UCI Credit Card Default

HarvardX (PH125.9x) - Data Science: Capstone

Wayne Koong Wah Yan

10/26/2020

1 Abstract

1.1 Purpose of this project

This project is the final report for my completion of HarvardX Data Science - Capstone course.

1.2 Objective of this project

The objective of this project is to apply the knowledge acquired throughout the Data Science courses. I am applying machine learning techniques toward the UCI Credit Card Default dataset obtained from kaggle.com.

1.3 "Too many cooks spoil the broth"

Interesting enough, while executing machine learning algorithms on the training dataset, evaluating using test dataset, I notice f-score and almost all other metrics value reduced if all variables in used. Better F-score achieved by using only selected variables.

Also another interesting finding, by including **LIMIT_BAL** and excluding all other **BILL_** and **PAY_** monetary variables, I have the best accuracy and f-score.

2 Introduction

Credit Card has becomes a necessity of lifestyle. Many financial institutions are relying on the credit card usage and payment history as a way to predict credit score for a person financial worthiness. For most financial institutions, Credit Card is the biggest contribution to the revenue and income, eclipse revenue from other businesses. Credit Card business, however, is high risk and unsecured, which sometimes, causing confusion within the financial institution: a bank branch manager has the authority to approve a 1,000,000 loan, but has no say on the issue of credit card with 100,000 credit limit.

The income from late payment are very fat, starting from 16% on outstanding amount. This reward is parallel with risk associated. Outstanding and default are closely monitored. The probability of default is the key assessment criteria during card issuance, thus, relying on historical dataset.

2.1 Recommendation Systems

Recommendation systems have been a big in the IT businesses since 80's. In 80's, it's known as Decision Support System, powered by defined/known algorithms suggesting probability of an event, whether categorical or continuous for management decision. Today, the Recommendation Systems are no longer solely relying on defined/known algorithms, but on continuous learning using machine learning and deep learning. The new form of Recommendation systems are made possible by the available of more powerful machines, with increased capability, scalability and affordability that allows application in wider scale.

Recommendation Systems is highly rely on given historical dataset that it's used to "train" the model. It learns from the past, and predict the future output either in categorical or continuous outcome. The output is typically in a predfiend scale, e.g. from 0 to 1, of which is subsequently interpreted into rating or preferences of choice, e.g. rating from 0 to 5, or choice of Yes or No, etc.

2.2 The UCI Credit Card Default Dataset

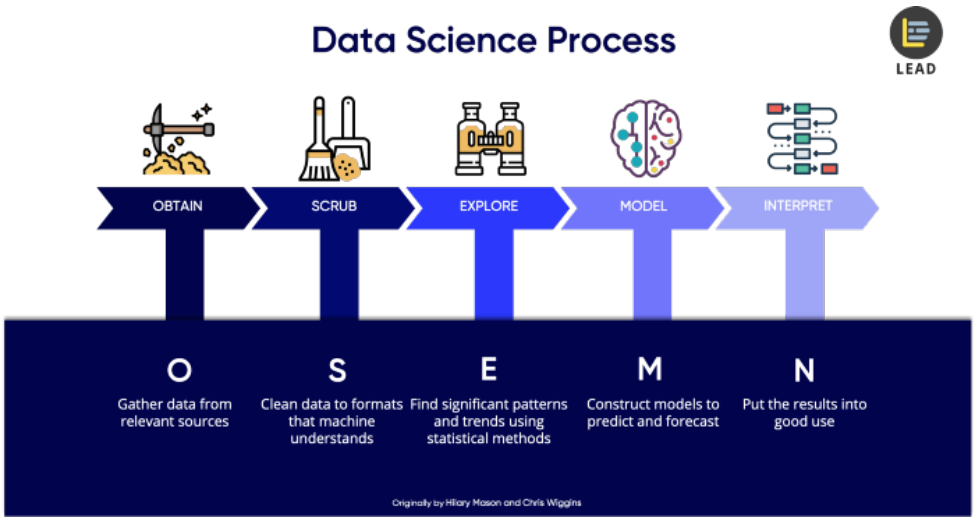
The UCI Credit Card Default dataset contains credit card clients billing and payment status of an organisation from Taiwan. This dataset contains information on default payments, demographic factors, credit data, history of payment, and bill statements for period from **April 2005 to September 2005**.

2.3 Goal

The aim of this project is to find the best prediction algorithm which can help financial institution in predicting a likelihood of a client default payment. I explored various machine learning algorithms, like linear regression, logistic regression, local regression (loess), random forest, gradient boost etc. The **Generalized Linear Model** is found to be the highest among all with **F-score value of 89.59%**.

2.4 Key steps

To ensure a standardized structural approach in organising and performing the analysis and prediction, I chosen and followed OSEMN Framework process.



OSEMN, the data science taxonomy (proposed by Hilary Mason and Chris Wiggins) or Snice taxonomy is the chronological order of Obtain, Scrub, Explore, Model, and iNterpret. This is a standardized process governs steps taken, ensuring probability of accurate analysis whilst allows steps to be backtrack-able and re-executable.

OSEMN Framework process is consists of following 5 steps:

- O - Obtaining our data
- S - Scrubbing / Cleaning our data
- E - Exploring / Visualizing our data will allow us to find patterns and trends
- M - Modeling our data will give us our predictive power as a wizard
- N - Interpreting our data

2.4.1 Step1 Obtain Data

Obtain data means identifying and acquiring the needed and correct dataset. This step is very important as a flawed dataset will miss-led and impact the probability of getting accurate prediction. After the complete dataset is obtained or downloaded, the data should be parsed and prepared in a form that is processable.

2.4.2 Step2 Scrub Data

Scrub data means to clean or filter unwanted "noise" from dataset. Depends on the quality of dataset, sometime a massive data cleansing may be required. This step needs to be executed properly and data should be explored from all angles, as "garbage in, garbage out" philosophy, not clean data with irrelevant or incorrectly parse data may rendered analyzed result null.

2.4.3 Step3 Explore Data

Explore data means examine data, or making sense of the dataset. This step involves careful data properties, e.g. data cardinally, relationship, factors, and data types like numerical, categorical, ordinal, nominal, etc. inspections.

Descriptive statistics are always compute to extract features and to test significant variables and their correlation. These extracted info are normally present in visualisation (e.g. chart) for patterns and trends identification.

2.4.4 Step4 Model Data

Model data is the step where models are selected, applied, tuned and executed to get the required outcome. This is the key step that resulted whether we able to produce a correct and high probability prediction, or a biased or wrong analysis.

Here, dimension of dataset is scrutinized and reduced if necessary, selecting the relevant features and value that contributes to the prediction of results. Various models are select and train:

- logistic regressions to perform classification to differentiate value,
- linear regression to forecast value,
- clustering by grouping data for better understanding of the logic,
- etc.

In short, regressions and predictions are typically use for forecasting future values, whilst classifications are to identify, and clustering to group values.

2.4.5 Step5 Interpreting Data

Interpreting Data means interpreting models and results, and presenting them in a human readable format. There is no standard format on how outcome should be presented. It can be simplified charts as those in newspaper, or series of highly technical charts for technical reader. A well-structured, clear and with actionable story report with relevant visual and data helps readers read and understands.

Regardless how good all other steps are performed, failure to present and communicate to the reader clearly & precisely means the efforts may not be appreciated (a.k.a getting continuous support, buy-in & funding).

3 Metrics & Methods

3.1 Metrics

Several metrics are studied to evaluate the best to present my prediction output. As the output is categorical, Confusion Matrix forms the base of the evaluation.

3.1.1 Confusion Matrix

Confusion matrix, or error matrix, is a table with 4 values that presents the outcome of prediction against actual result. It gives a visualisation of the performance of an algorithm, with each row of the matrix represents the predicted class while each column represents the actual class.

- True positive (TP): Positive outcome correctly identified as positive,
- False positive (FP): Positive outcome incorrectly identified as negative,
- True negative (TN): Negative outcome correctly identified as negative, and
- False negative (FN): Negative outcome incorrectly identified as positive.

Table 1: CONFUSION MATRIX

	Actual Positive(1)	Actual Negative(0)
Predicted Positive(1)	TP	FN
Predicted Negative(0)	FP	TN

3.1.2 Accuracy

Accuracy is defined as how close the predicted positive outcome over actual positive outcome, and the predicted negative outcome over the actual negative outcome:

$$Accuracy = \frac{TP + TN}{N}$$

when N is total populations.

3.1.3 Sensitivity/Recall

sensitivity, or Recall is defined as the ability of an algorithm to predict a positive outcome when the actual outcome is positive: $\hat{Y} = 1$ when $Y = 1$. Thus

$$High\ Sensitivity:\ Y = 1 ==> \hat{Y} = 1$$
$$Sensitivity = \frac{TP}{(TP + FN)}$$

3.1.4 Specificity/Precision

Specificity, or Precision is defined as the ability of an algorithm to not predict a positive $\hat{Y} = 0$ when the actual outcome is not a positive $Y = 0$. Thus

$$High\ Specificity:\ Y = 0 ==> \hat{Y} = 0$$

$$Specificity = \frac{TN}{(TN + FP)}$$

Specificity also can defined as the proportion of positive calls that are actually positive

$$High\ Specificity:\ \hat{Y} = 1 ==> hatY = 1$$
$$Specificity = \frac{TP}{(TP + FP)}$$

3.1.5 Prevalence

Prevalence is the number of actual positive at a given time, vs the actual negative. Thus, when N is the total population:

$$Prevalence\ (Negative):\ \frac{Y = 1}{N}$$
$$Prevalence\ (Negative):\ \frac{Y = 0}{N}$$

Due to imbalance prevalence toward a class, an overall high accuracy is possible despite relatively low sensitivity or low specificity. Thus, Balanced Accuracy and F-Score should be examined on imbalance prevalence case.

3.1.6 F-Score

F-score or F-measure is a measure of a test's accuracy. It is the harmonic mean of the precision and recall. The highest possible value of an F-score is 1, indicating perfect precision and recall, and the lowest possible value is 0, if either the precision or the recall is zero. The F1 score is also known as the Sørensen–Dice coefficient or Dice similarity coefficient (DSC). The formula of F-Score, F_1 is:

$$F_1 = 2 \times \frac{Sensitivity \times Specificity}{Sensitivity + Specificity}$$

3.2 Method

Method selections is a major criteria in my report. Many methods are available, from caret package, and other packages. I refer to teaching materials, and some browsing, selected following methods to train and predict the UCI Credit Card Default. Selected methods are:

- Generalized Linear Model,
- Linear Model,
- Generalized Additive Model using LOESS
- K-nearest Neighbors
- K-Means Clustering
- Random Forest
- Gradient Boosting

3.2.1 Generalized Linear Model

Generalized linear model (GLM) is a flexible generalization of ordinary linear regression that allows for response variables that have error distribution models other than a normal distribution. The GLM generalizes linear regression by allowing the linear model to be related to the response variable via a link function and by allowing the magnitude of the variance of each measurement to be a function of its predicted value.

In a generalized linear model (GLM), each outcome Y of the dependent variables is assumed to be generated from a particular distribution in an exponential family, a large class of probability distributions that includes the normal, binomial, Poisson and gamma distributions, among others. The mean, μ, of the distribution depends on the independent variables, X, through:

$$E(Y) = \mu = g^{-1}(X\beta)$$

where $E(Y)$ is the expected value of Y; $X\beta$ is the linear predictor, a linear combination of unknown parameters β; g is the link function.

In this framework, the variance is typically a function, V, of the mean:

$$Var(Y) = V(\mu) = V(g^{-1}(X\beta)$$

It is convenient if V follows from an exponential family of distributions, but it may simply be that the variance is a function of the predicted value. The unknown parameters, β, are typically estimated with maximum likelihood, maximum quasi-likelihood, or Bayesian techniques.

3.2.2 Linear Model

LInear Model, or linear regression model, is a linear approach to modeling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). The relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data, calculating the conditional mean of the response given the values of the explanatory variables (or predictors). Linear regression focuses on the conditional probability distribution of the response given the values of the predictors, rather than on the joint probability distribution of all of these variables.

Given a data set $\{y_i, x_{i1}, ..., x_{iy}\}_{i=1}^{n}$ of n statistical units, a linear regression model assumes that the relationship between the dependent variable y and the p-vector of regressors x is linear. This relationship is modeled through a disturbance term or error variable ϵ — an unobserved random variable that adds "noise" to the linear relationship between the dependent variable and regressors. Thus the model takes the form

$$y_i = \beta_0 + \beta_1 x_{i1} + ... + \beta_y X_{ip} + \epsilon_i = x_i^T \beta + \epsilon_i$$

where $i = 1, ..., n$, and ^{T} means transpose, so that $x_i^T \beta$ is the inner product between vectors x_i and β.

3.2.3 Generalized Additive Model using LOESS

Generalized Additive Model or GAM is an additive modeling technique where the impact of the predictive variables is captured through smooth functions which—depending on the underlying patterns in the data—can be nonlinear. GAM is writes in the form:

$$g(E(Y)) = \alpha + s_1(x_1) + ... + s_p(x_p)$$

where Y is the dependent variable, $E(Y)$ denotes the expected value, and $g(Y)$ denotes the link function that links the expected value to the predictor variables $x_1, \ldots, x_p$. The terms $s_1(x_1), \ldots, s_p(x_p)$ denote smooth, nonparametric functions.

LOESS (locally estimated scatterplot smoothing) is method developed for calculation of local regression, local polynomial regression or moving regression, which is a generalization of moving average and polynomial regression. LOESS is builds on "classical" methods, such as linear and nonlinear least squares regression. It address situations in which the classical procedures do not perform well or cannot be effectively applied without undue labor. LOESS combines much of the simplicity of linear least squares regression with the flexibility of nonlinear regression. It does this by fitting simple models to localized subsets of the data to build up a function that describes the deterministic part of the variation in the data, point by point. In fact, one of the chief attractions of this method is that the data analyst is not required to specify a global function of any form to fit a model to the data, only to fit segments of the data.

3.2.4 K-nearest Neighbors

K-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression, where

- for k-NN classification, an object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. If $k = 1$, then the object is simply assigned to the class of that single nearest neighbor.
- for k-NN regression, the output value is the average of the values of k nearest neighbors.

K-NN for categorical variables use Hamming distance functions as the form below

$$Hamming\ Distance = D_H = \sum_{i=1}^{k} |x_i - y_i|$$

Where,

$$x = y \ => \ D = 0$$
$$x \neq y \ => D = 1$$

3.2.5 K-Means Clustering

k-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells.

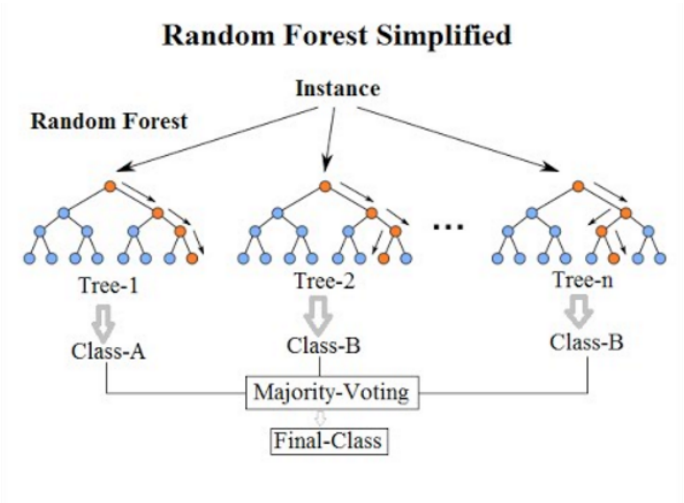
Given a set of observations $(x_1, x_2, ..., x_n)$, where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into $k(\leq n)$ sets $S = S_1, S_2, ..., S_k$ so as to minimize the within-cluster sum of squares:

$$arg\ min \sum_{i=1}^{k} \sum_{x \epsilon S_i} ||x - \mu||^2 = arg\ min \sum_{i=1}^{k} |S_i| VarS_i$$

where μ_i is the mean of points in S_i.

3.2.6 Random Forest

Random forests is an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean/average prediction (regression) of the individual trees. Random forests correct for decision trees' habit of overfitting to their training set and generally outperform decision trees, but lower than gradient boosted trees.



Random forests are frequently used as "blackbox" models in businesses, and involves following steps/algorithms:

Decision Tree Learning
Decision tree learning is one of the predictive modelling approaches. It uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees.

Forests are like the pulling together of decision tree algorithm efforts. Taking the teamwork of many trees thus improving the performance of a single random tree. Though not quite similar, forests give the effects of a K-fold cross validation.

Bagging
Bagging, or bootstrap aggregating, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It also reduces variance and helps to avoid overfitting.

3.2.7 Gradient Boosting

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

XGBoost package is used for Gradient Boosting estimation, compare with other gradient boosting algorithms, XGBoost:

- Clever penalization of trees
- A proportional shrinking of leaf nodes
- Newton Boosting
- Extra randomization parameter
- Implementation on single, distributed systems and out-of-core computation

4 Obtain Data

4.1 Data Source

The UCI Credit Card Default dataset is download from Kaggle.com using following URL:

- https://www.kaggle.com/uciml/default-of-credit-card-clients-dataset

4.2 Dataset Dimension

The downloaded dataset consists of 25 columns and 30,000 of records.

Table 2: DATASET DIMENION

	Count
Row	30,000
Column	25

4.3 Variables

The obtained dataset contains 25 variables, which are:

Table 3: VARIABLES DEFINITION

ID	ID of each client
LIMIT_BAL	Credit Limit, Amount of given credit in NT dollars (includes individual and family/supplementary credit
SEX	Gender (1=male, 2=female)
EDUCATION	Education (0=?, 1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)
MARRIAGE	Marital status (0=?,1=married, 2=single, 3=others)
AGE	Age in years
PAY_0	Repayment status in September, 2005 (-1=pay duly, 1=payment delay for one month, 2=payment delay for two months, ... 8=payment delay for eight months, 9=payment delay for nine months and above)
PAY_2	Repayment status in August, 2005 (scale same as above)
PAY_3	Repayment status in July, 2005 (scale same as above)
PAY_4	Repayment status in June, 2005 (scale same as above)
PAY_5	Repayment status in May, 2005 (scale same as above)
PAY_6	Repayment status in April, 2005 (scale same as above)
BILL_AMT1	Amount of bill statement in September, 2005 (NT dollar)
BILL_AMT2	Amount of bill statement in August, 2005 (NT dollar)
BILL_AMT3	Amount of bill statement in July, 2005 (NT dollar)
BILL_AMT4	Amount of bill statement in June, 2005 (NT dollar)
BILL_AMT5	Amount of bill statement in May, 2005 (NT dollar)
BILL_AMT6	Amount of bill statement in April, 2005 (NT dollar)
PAY_AMT1	Amount of previous payment in September, 2005 (NT dollar)
PAY_AMT2	Amount of previous payment in August, 2005 (NT dollar)
PAY_AMT3	Amount of previous payment in July, 2005 (NT dollar)
PAY_AMT4	Amount of previous payment in June, 2005 (NT dollar)
PAY_AMT5	Amount of previous payment in May, 2005 (NT dollar)

Table 3: VARIABLES DEFINITION

PAY_AMT6	Amount of previous payment in April, 2005 (NT dollar)
default.payment.next.month	Default payment (1=yes, 0=no)

4.4 NA Value

The last data cleansing is to ensure no NA data that will significantly affecting modeling exercise. Based on below function output, there is no NA data in the dataset.

```
anyNA(dat_raw)
```

```
[1] FALSE
```

4.5 Variables Summary

Following are preliminary analysis of all 25 variables:

```
### Data Frame Summary
#### dat_raw
**Dimensions:** 30000 x 25
**Duplicates:** 0

+----+------------------------------+-----------------------------------+-------------------------+
| No | Variable                     | Stats / Values                    | Freqs (% of Valid)      |
+====+==============================+===================================+=========================+
| 1  | ID\                          | Mean (sd) : 15000.5 (8660.4)\     | 30000 distinct values\  |
|    | [integer]                    | min < med < max:\                 | (Integer sequence)      |
|    |                              | 1 < 15000.5 < 30000\              |                         |
|    |                              | IQR (CV) : 14999.5 (0.6)          |                         |
+----+------------------------------+-----------------------------------+-------------------------+
| 2  | LIMIT_BAL\                   | Mean (sd) : 167484.3 (129747.7)\  | 81 distinct values      |
|    | [numeric]                    | min < med < max:\                 |                         |
|    |                              | 10000 < 140000 < 1000000\         |                         |
|    |                              | IQR (CV) : 190000 (0.8)           |                         |
+----+------------------------------+-----------------------------------+-------------------------+
| 3  | SEX\                         | Min  : 1\                         | 1 : 11888 (39.6%)\      |
|    | [integer]                    | Mean : 1.6\                       | 2 : 18112 (60.4%)       |
|    |                              | Max  : 2                          |                         |
+----+------------------------------+-----------------------------------+-------------------------+
| 4  | EDUCATION\                   | Mean (sd) : 1.9 (0.8)\            | 0 :    14 ( 0.0%)\      |
|    | [integer]                    | min < med < max:\                 | 1 : 10585 (35.3%)\      |
|    |                              | 0 < 2 < 6\                        | 2 : 14030 (46.8%)\      |
|    |                              | IQR (CV) : 1 (0.4)                | 3 :  4917 (16.4%)\      |
|    |                              |                                   | 4 :   123 ( 0.4%)\      |
|    |                              |                                   | 5 :   280 ( 0.9%)\      |
|    |                              |                                   | 6 :    51 ( 0.2%)       |
+----+------------------------------+-----------------------------------+-------------------------+
| 5  | MARRIAGE\                    | Mean (sd) : 1.6 (0.5)\            | 0 :    54 ( 0.2%)\      |
|    | [integer]                    | min < med < max:\                 | 1 : 13659 (45.5%)\      |
|    |                              | 0 < 2 < 3\                        | 2 : 15964 (53.2%)\      |
```

		IQR (CV) : 1 (0.3)	3 : 323 (1.1%)
6	AGE\ [integer]	Mean (sd) : 35.5 (9.2)\ min < med < max:\ 21 < 34 < 79\ IQR (CV) : 13 (0.3)	56 distinct values
7	PAY_0\ [integer]	Mean (sd) : 0 (1.1)\ min < med < max:\ -2 < 0 < 8\ IQR (CV) : 1 (-67.3)	11 distinct values
8	PAY_2\ [integer]	Mean (sd) : -0.1 (1.2)\ min < med < max:\ -2 < 0 < 8\ IQR (CV) : 1 (-8.9)	11 distinct values
9	PAY_3\ [integer]	Mean (sd) : -0.2 (1.2)\ min < med < max:\ -2 < 0 < 8\ IQR (CV) : 1 (-7.2)	11 distinct values
10	PAY_4\ [integer]	Mean (sd) : -0.2 (1.2)\ min < med < max:\ -2 < 0 < 8\ IQR (CV) : 1 (-5.3)	11 distinct values
11	PAY_5\ [integer]	Mean (sd) : -0.3 (1.1)\ min < med < max:\ -2 < 0 < 8\ IQR (CV) : 1 (-4.3)	-2 : 4546 (15.2%)\ -1 : 5539 (18.5%)\ 0 : 16947 (56.5%)\ 2 : 2626 (8.8%)\ 3 : 178 (0.6%)\ 4 : 84 (0.3%)\ 5 : 17 (0.1%)\ 6 : 4 (0.0%)\ 7 : 58 (0.2%)\ 8 : 1 (0.0%)
12	PAY_6\ [integer]	Mean (sd) : -0.3 (1.1)\ min < med < max:\ -2 < 0 < 8\ IQR (CV) : 1 (-4)	-2 : 4895 (16.3%)\ -1 : 5740 (19.1%)\ 0 : 16286 (54.3%)\ 2 : 2766 (9.2%)\ 3 : 184 (0.6%)\ 4 : 49 (0.2%)\ 5 : 13 (0.0%)\ 6 : 19 (0.1%)\ 7 : 46 (0.1%)\ 8 : 2 (0.0%)
13	BILL_AMT1\ [numeric]	Mean (sd) : 51223.3 (73635.9)\ min < med < max:\ -165580 < 22381.5 < 964511\ IQR (CV) : 63532.2 (1.4)	22723 distinct values

14	BILL_AMT2\ [numeric]	Mean (sd) : 49179.1 (71173.8)\ min < med < max:\ -69777 < 21200 < 983931\ IQR (CV) : 61021.5 (1.4)	22346 distinct values	
15	BILL_AMT3\ [numeric]	Mean (sd) : 47013.2 (69349.4)\ min < med < max:\ -157264 < 20088.5 < 1664089\ IQR (CV) : 57498.5 (1.5)	22026 distinct values	
16	BILL_AMT4\ [numeric]	Mean (sd) : 43262.9 (64332.9)\ min < med < max:\ -170000 < 19052 < 891586\ IQR (CV) : 52179.2 (1.5)	21548 distinct values	
17	BILL_AMT5\ [numeric]	Mean (sd) : 40311.4 (60797.2)\ min < med < max:\ -81334 < 18104.5 < 927171\ IQR (CV) : 48427.5 (1.5)	21010 distinct values	
18	BILL_AMT6\ [numeric]	Mean (sd) : 38871.8 (59554.1)\ min < med < max:\ -339603 < 17071 < 961664\ IQR (CV) : 47942.2 (1.5)	20604 distinct values	
19	PAY_AMT1\ [numeric]	Mean (sd) : 5663.6 (16563.3)\ min < med < max:\ 0 < 2100 < 873552\ IQR (CV) : 4006 (2.9)	7943 distinct values	
20	PAY_AMT2\ [numeric]	Mean (sd) : 5921.2 (23040.9)\ min < med < max:\ 0 < 2009 < 1684259\ IQR (CV) : 4167 (3.9)	7899 distinct values	
21	PAY_AMT3\ [numeric]	Mean (sd) : 5225.7 (17607)\ min < med < max:\ 0 < 1800 < 896040\ IQR (CV) : 4115 (3.4)	7518 distinct values	
22	PAY_AMT4\ [numeric]	Mean (sd) : 4826.1 (15666.2)\ min < med < max:\ 0 < 1500 < 621000\ IQR (CV) : 3717.2 (3.2)	6937 distinct values	
23	PAY_AMT5\ [numeric]	Mean (sd) : 4799.4 (15278.3)\ min < med < max:\ 0 < 1500 < 426529\ IQR (CV) : 3779 (3.2)	6897 distinct values	
24	PAY_AMT6\ [numeric]	Mean (sd) : 5215.5 (17777.5)\ min < med < max:\ 0 < 1500 < 528666\ IQR (CV) : 3882.2 (3.4)	6939 distinct values	

```
+----+------------------------------+----------------------------------+------------------------+
| 25 | default.payment.next.month\ | Min  : 0\                        | 0 : 23364 (77.9%)\     |
|    | [integer]                    | Mean : 0.2\                      | 1 :  6636 (22.1%)      |
|    |                              | Max  : 1                         |                        |
+----+------------------------------+----------------------------------+------------------------+
```

5 Scrub Data

5.1 ID

Variable ID is the unique identifier of the dataset and play no role in analysis. It's useful for unique identically a row of record, but will mislead subsequent modeling efforts. Thus, I removed this variable from the main dataset.

```
dat_raw <- dat_raw[,-1]
```

5.2 default.payment.next.month

Variable default.payment.next.month refers to whether a client has defaulted Oct 2005 credit card payment. Default in credit card payment means failure to make a payment by the due date. If this happens, the credit card issuance financial institution will impose penalty charge to the default, and if the case has become delinquency, legal action will be taken to enforce payment.

This variable name is kind of too long and not in consistent naming convention of other variables, thus, I renamed the variable to DEFAULT.

```
names(dat_raw)[24] <- "DEFAULT"
```

5.3 PAY_0

Variable PAY_0 is defined as "Repayment status in September, 2005 (-1=pay duly, 1=payment delay for one month, 2=payment delay for two months, ... 8=payment delay for eight months, 9=payment delay for nine months and above)", which is similar with variables PAY_2, PAY_3, PAY_4, PAY_5 and PAY_6. I renamed the column PAY_0 to PAY_1 to made it consistent with variables BILL_AMT1 & PAY_AMT1 which means value for September 2005.

```
names(dat_raw)[6] <- "PAY_1"
```

5.4 PAY_AMTx and BILL_AMTx

Variables PAY_AMT1, PAY_AMT2 ... PAY_AMT6 which refers to "Amount of previous payment" and BILL_AMT1, BILL_AMT2 ... BILL_AMT6 which refers to "Amount of bill statement" are kind of confusion when refer together with PAY_1, PAY_2 ... PAY_6. For ease of references throughout the report and modeling, I renamed these columns respectively

- BILL_AMTx to BILLED_x
- PAY_AMTx to PAYMENT_x

```
colnames(dat_raw)[12:17] <- rep(paste("BILLED",1:6,sep="_"),1)
colnames(dat_raw)[18:23] <- rep(paste("PAYMENT",1:6,sep="_"),1)
```

5.5 FACTORIZED DATA

Refers to above variables definition and variables summary, following variables are identified as categorical data that should be factorized:

Table 4: CATEGORICAL VARIABLES

SEX	Gender (1=male, 2=female)
EDUCATION	Education (0=?, 1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)
MARRIAGE	Marital status (0=?,1=married, 2=single, 3=others)
DEFAULT	Default payment (1=yes, 0=no)

These 4 variables are factorized with following syntax:

```
tmp_categorical <- c('SEX','EDUCATION','MARRIAGE','DEFAULT')
dat_raw[tmp_categorical] <- lapply(dat_raw[tmp_categorical],
                                   function(x) as.factor(x))
rm(tmp_categorical)
```

Apart of above 4 variables, PAY_1, PAY_2, PAY_3, PAY_4, PAY_5 & PAY_6 are also categorical, however I do not factorized these variables due to analysis that to be performed later.

5.6 SEX

As defined in variable definition, SEX is categorical data with 2 value,

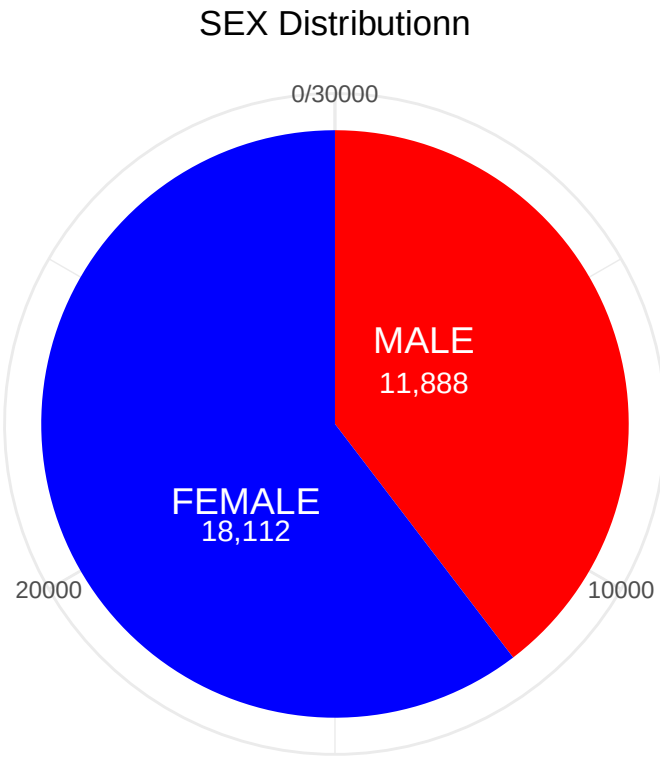
- 1 means male, and
- 2 means female.

Checking levels of SEX data, the data is found to be comformed with the above definition thus no further actio required.

Table 5: EDUCATION VARIABLE

1	2
11,888	18,112

By plotting SEX Distribution, there are total of 18,112 FEMALE vs 11,888 MALE. The prevalence of SEX data is slightly imbalance by FEMALE over MALE of 152.36%.



5.7 EDUCATION

As defined in variable definition, EDUCATION is categorical data with 6 value, which respectively means:

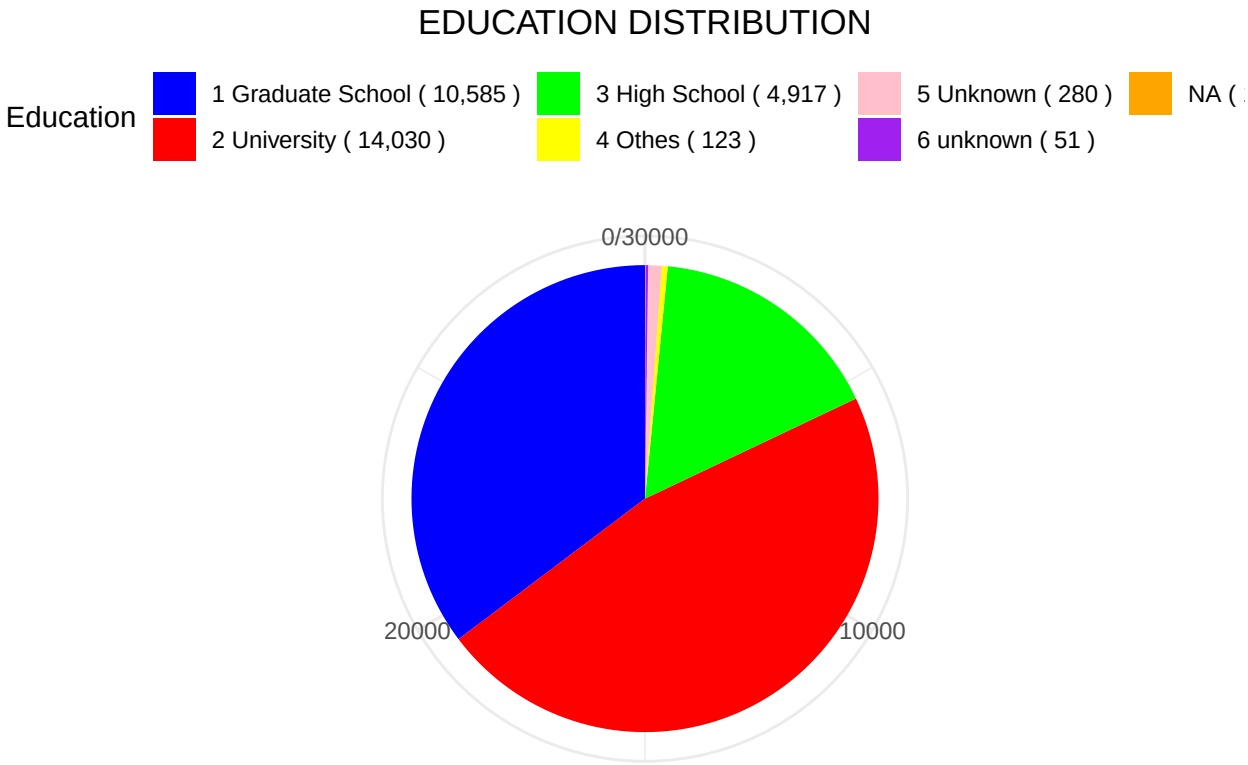
- 1=graduate school,
- 2=university,
- 3=high school,
- 4=others,
- 5=unknown, and
- 6=unknown.

The class 4, 5 & 6 are all by definition refers to the same definition. Although it may be interpreted differently by the source owner and influence modeling output, however, by retains these 3 classes as current form will impact my interpretation of the result. Thus, I am going to modify class 4, 5 & 6 to class 4.

Before executing the data parsing, I examined the data cardinally:

Table 6: EDUCATION VARIABLE

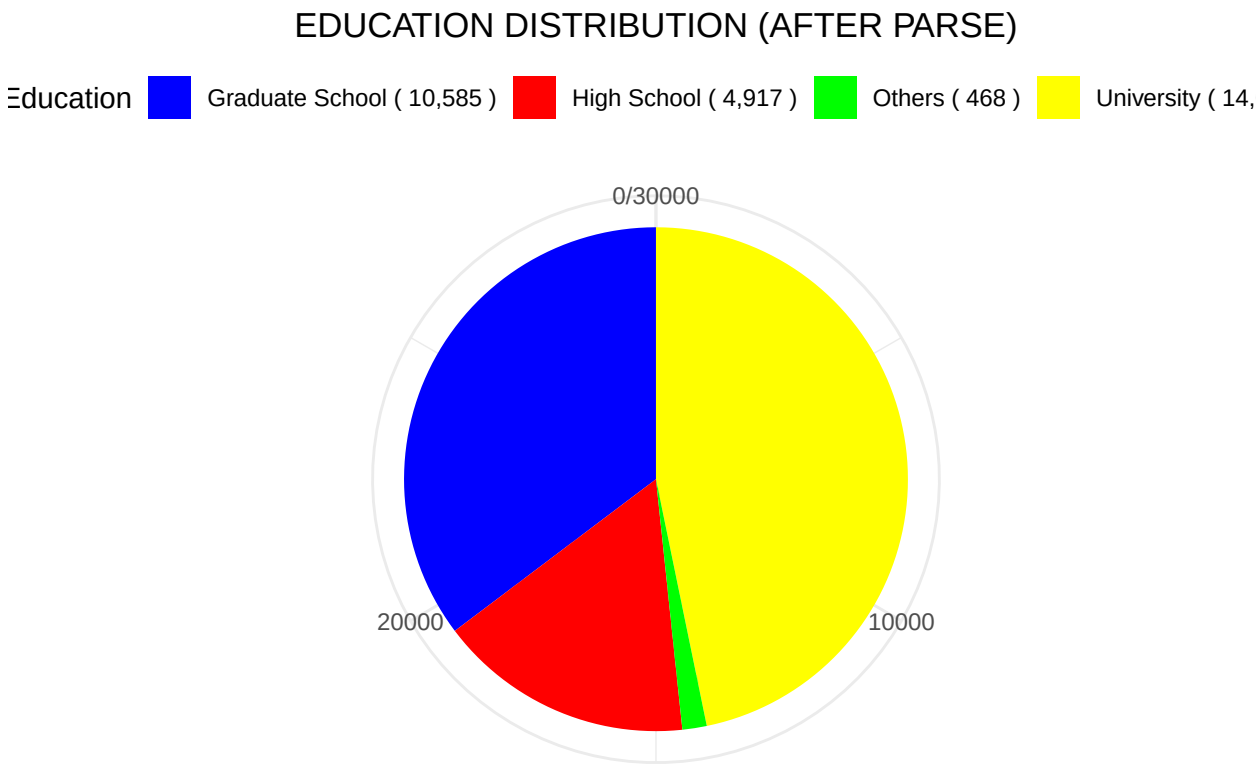
0	1	2	3	4	5	6
14	10,585	14,030	4,917	123	280	51



Surprisingly, there is a class 0 that's not defined, that should be also classified as class 4 "others". Following codes are executed to parse Education data:

```
dat_raw$EDUCATION[which(dat_raw$EDUCATION %in% c(0,4,5,6))] = 4
```

From the below parsed EDUCATION variable distribution, most clients (46.7667%) are having University level education.



5.8 MARRIAGE

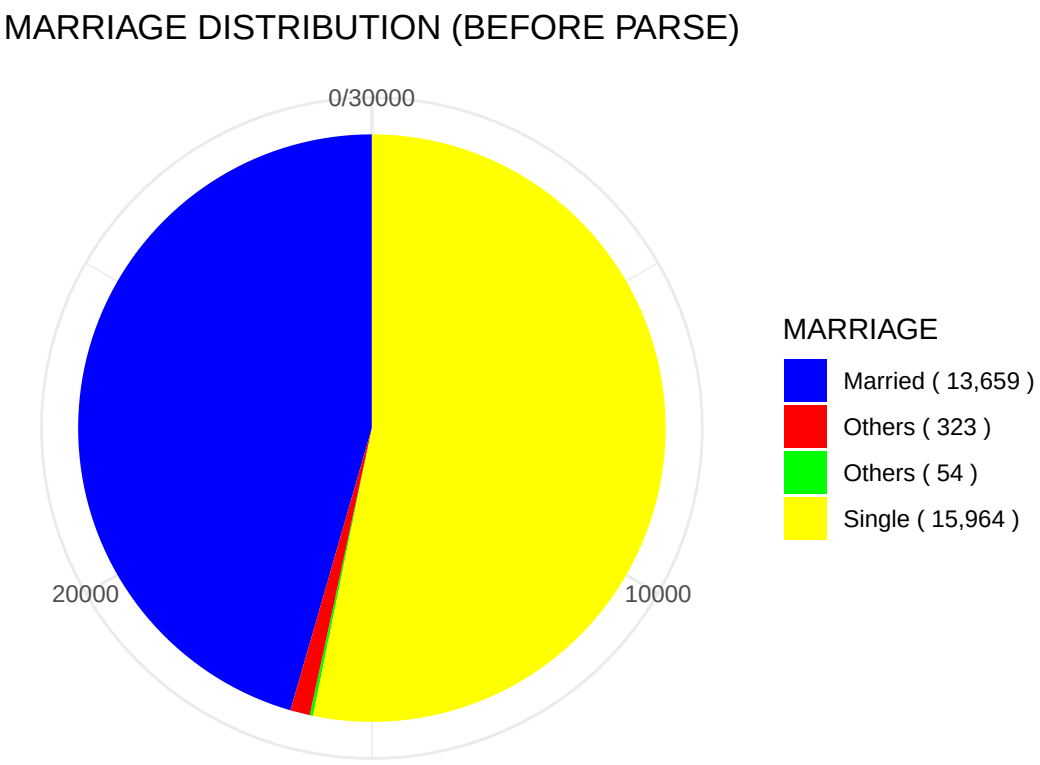
Based on variable definition, MARRIAGE is categorical data with 3 values:

- 1=Married,
- 2=Single, and
- 3=Others.

However, based on data cardinally examination, there is Class 0 that is not defined:

Table 7: MARRIAGE VARIABLE

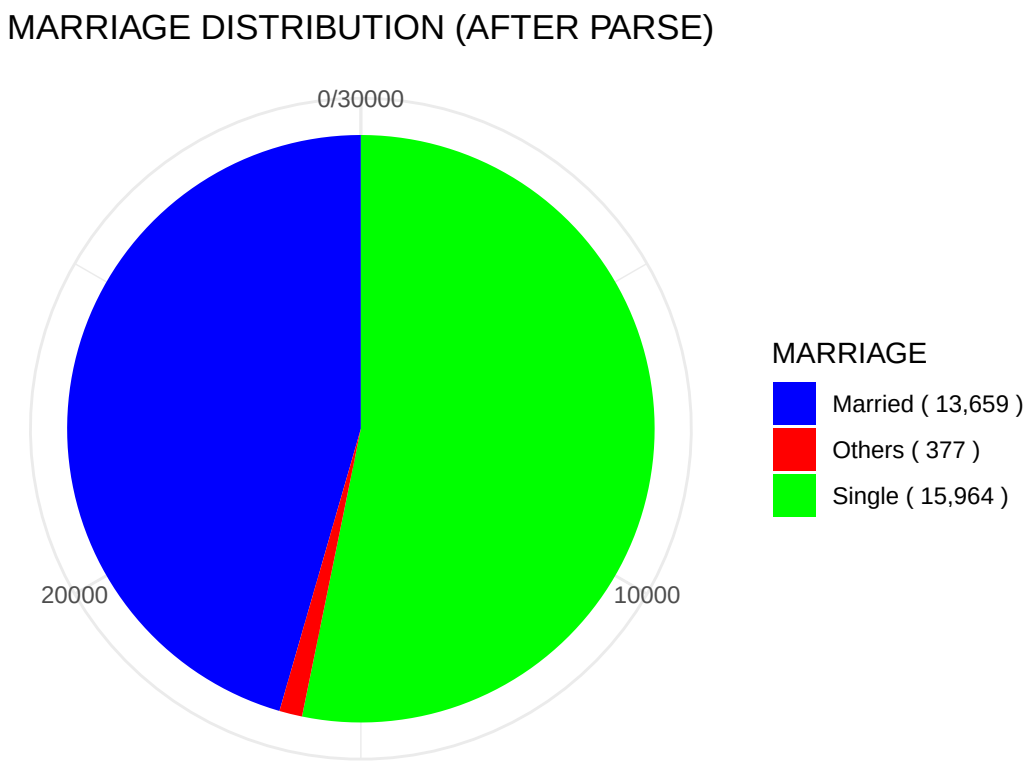
0	1	2	3
54	13,659	15,964	323



The Class 0 is parsed to Class 3 Others, using following codes:

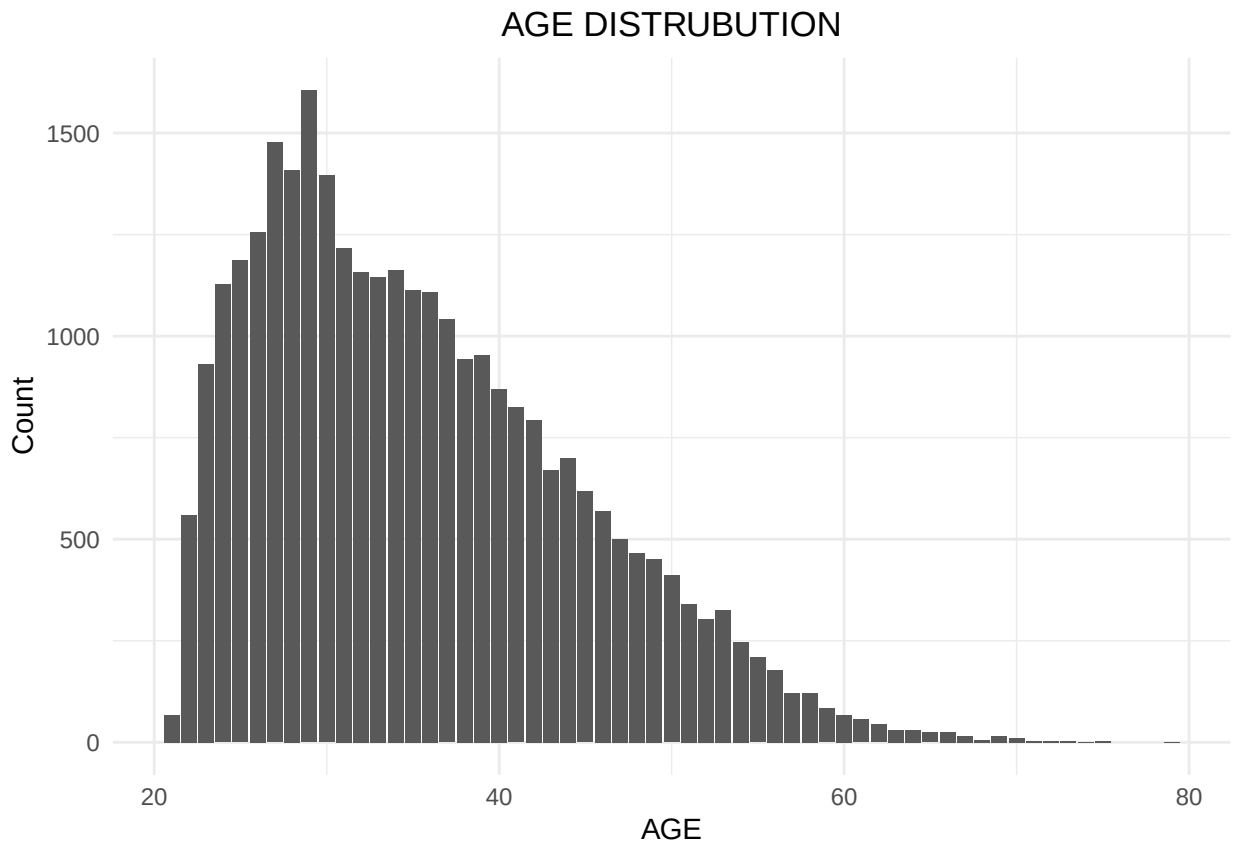
```
dat_raw$MARRIAGE[which(dat_raw$MARRIAGE %in% c(0,3))] = 3
```

From the below parsed MARRIAGE variable distribution, most clients (45.53%) are Single.



5.9 AGE

AGE variable refers to the age of client ranging from 21 to 79 with distribution as below. The AGE distribution is skewed right.



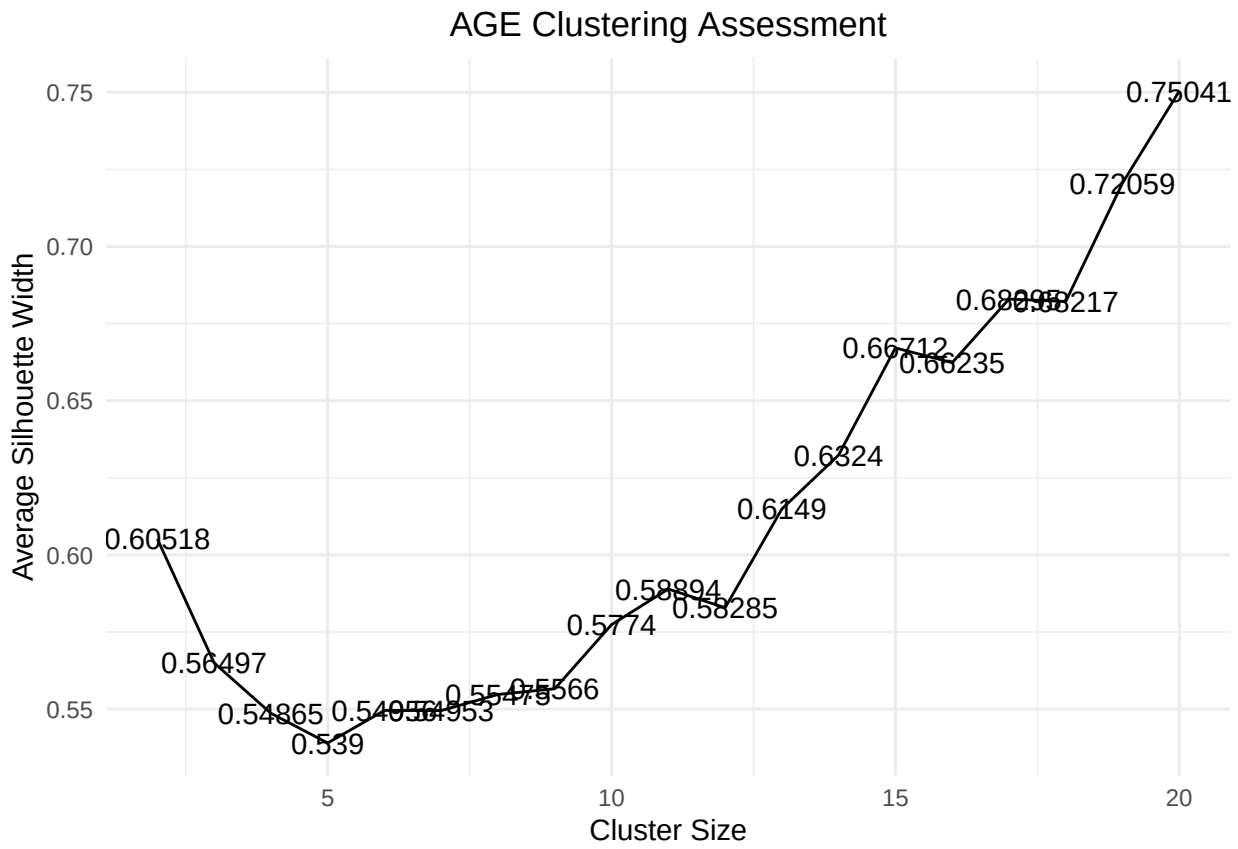
In order to minimized numbers of AGE class, AGE variable is grouped using Average Silhouette.

Silhouette analysis
Silhouette analysis is used to study the separation distance between the resulting clusters. The silhouette plot displays a measure of how close each point in one cluster is to points in the neighboring clusters and thus provides a way to assess parameters like number of clusters visually. Silhouette coefficients (as these values are referred to as) near +1 indicate that the sample is far away from the neighboring clusters. A value of 0 indicates that the sample is on or very close to the decision boundary between two neighboring clusters and negative values indicate that those samples might have been assigned to the wrong cluster.

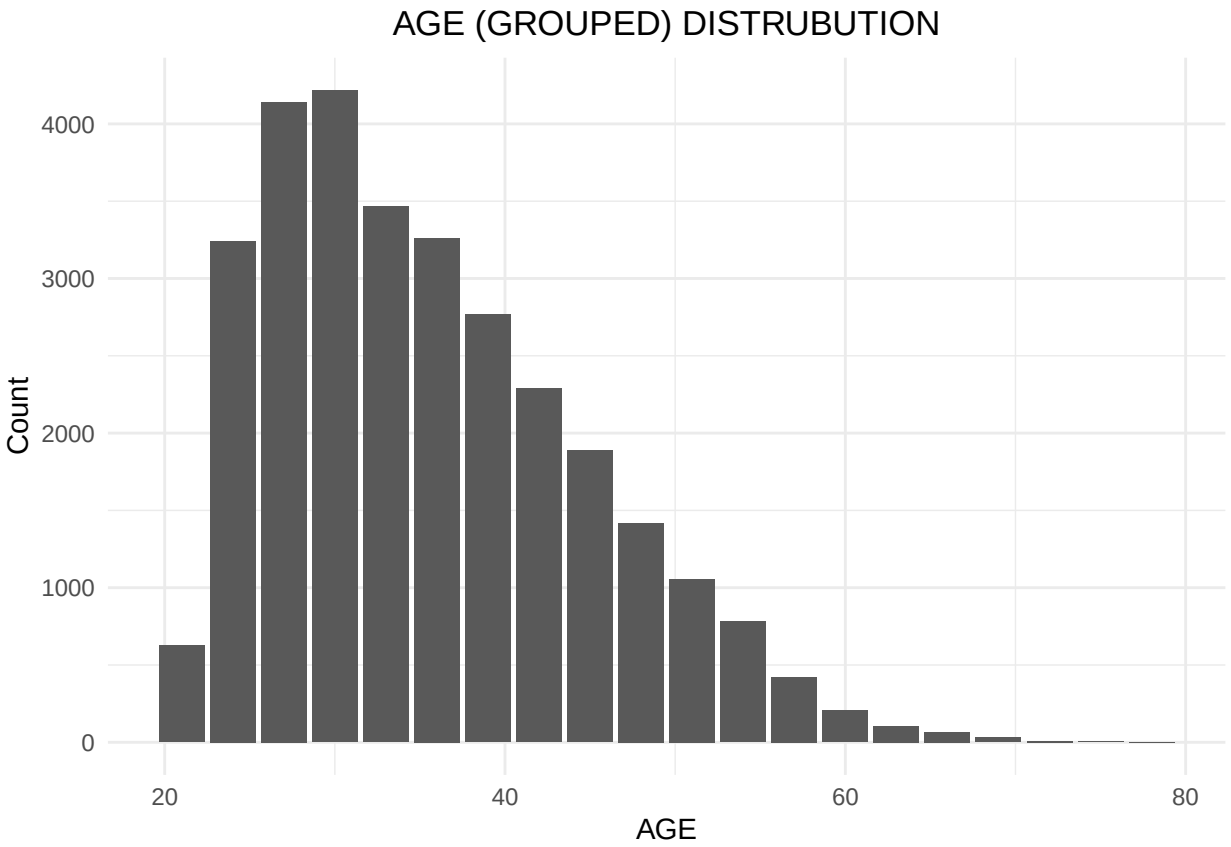
CAUTION
Processing of following codes will be taking more than **3 hours**. Do run only if necessary. The results from my execution is hardcoded for ease of reference in this report.

```
tmp_cluster <- numeric(20)
tmp_k <- seq(2,20)
tmp_k <- c(2,20)

tmp_cluster <- sapply(tmp_k, function(x){
  tmp_start_timer <- Sys.time()
  tmp_avg <- pam(dat_raw$AGE, x)$silinfo$avg.width
  cat("silhouette analysis for cluster:", x,
      ", average:",tmp_avg,"\n")
```

```
  cat("Now:", print(Sys.time(),"%d.%m.%Y %H:%M:%S"),", proc time:",
      print(difftime(tmp_start_timer,Sys.time())), "\n")
  tmp_avg
})
```



Based on above chart, the recommended clustering size using Silhouett Analysis is 20. Thus, it means that the original 59 classes should be reduced into 20, which translated into 3 years per class:


AGE (GROUPED) DISTRUBUTION
Count
4000
3000
2000
1000
0
20
40
60
80
AGE

5.10 BILLED_x & PAYMENT_x

According to variables definition provide d, BILLED_x are referring to billed credit card amount and PAYMENT_x are referring to actual credit card payment made for the month. In the credit card industry, a client can choose to pay minimal amount for the monthly bills, paying partial, paying full as billed, or paying more than billed. Another important info that is not explained is whether the BILLED_x amount is the actual billed or the part of the amount spent for the month:

$$BILLED\_x = (\sum BILLED_x + BILLED_{x-1} + ... + BILLED_{x-n}) -$$
$$(\sum PAYMENT_{x-1} + PAYMENT_{x-2} + ... + PAYMENT_{x-n})$$

or,

$$BILLED\_x = \sum ACTUAL\ SPENT_x$$

Cross referencing between PAYMENT_x and BILLED_x are useless without the definition is clarified. Also, by using credit card industry definition, BILLED amount can be negative, in which client paid more than billed amount in previous month, however, PAYMENT amount must be always positive. Thus, checking of any negative PAYMENT_x amount is necessary:

There are total of 0 PAYMENT_X that are negative, thus, no parse required.

5.11 LIMIT_BAL

Based on Credit Card Industry definition, LIMIT_BAL means the maximum amount a client can spent, which means the last billed amount should be always less than the limit balance, or a small % above the limit balance due to the monthly interest charge.

However, while cross checking between LIMIT_BAL and PAYMENT_1, PAYMENT_2, BILLED_1 & BILLED_2, the dataset presented is not conformed with the Credit Card Industry practice. For example, following are some sample data that are do not make sense:

Table 8: ABNORMAL DATA

LIMIT_BAL	DIFF_PERCENTAGE	BILLED_1	BILLED_2	PAYMENT_1	PAYMENT_2
20,000	5.4553	129,106	127,610	4,420	3,650
10,000	4.3095	53,095	54,562	2,134	1,000
20,000	2.7255	74,510	75,262	2,734	1,200
30,000	2.9529	118,587	89,271	6,624	3,000
20,000	3.1406	82,812	85,352	2,697	6,000
50,000	1.0560	102,800	0	0	0
100,000	0.5497	154,970	0	0	0
20,000	0.8454	36,908	0	0	0
20,000	1.1670	43,340	42,619	0	0
80,000	2.1449	251,594	264,594	13,000	0
150,000	0.9239	288,585	282,677	0	0

5.12 PAY_x

As describe in the variable definitions, the value of PAY_x means the number of month payment delayed. Therefore, for a record which PAY_1 = 8, means the client only made credit card payment after Jun 2006 (Oct 2005 + 9 months).

Table 9: SAMPLE PAY_x DATA

PAY_1	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6	PAID_1	PAID_2	PAID_3	PAID_4	PAID_5	PA
8	7	6	5	4	3	May 2006	Mar 2006	Jan 2006	Nov 2005	Sep 2005	Jul
8	7	6	5	4	3	May 2006	Mar 2006	Jan 2006	Nov 2005	Sep 2005	Jul
8	7	6	5	4	3	May 2006	Mar 2006	Jan 2006	Nov 2005	Sep 2005	Jul
8	7	6	5	4	3	May 2006	Mar 2006	Jan 2006	Nov 2005	Sep 2005	Jul
8	7	6	5	4	3	May 2006	Mar 2006	Jan 2006	Nov 2005	Sep 2005	Jul
8	7	6	5	4	3	May 2006	Mar 2006	Jan 2006	Nov 2005	Sep 2005	Jul
8	7	6	5	4	3	May 2006	Mar 2006	Jan 2006	Nov 2005	Sep 2005	Jul
8	7	6	5	4	3	May 2006	Mar 2006	Jan 2006	Nov 2005	Sep 2005	Jul
8	7	6	5	4	3	May 2006	Mar 2006	Jan 2006	Nov 2005	Sep 2005	Jul
8	7	6	5	4	3	May 2006	Mar 2006	Jan 2006	Nov 2005	Sep 2005	Jul
8	7	6	5	4	3	May 2006	Mar 2006	Jan 2006	Nov 2005	Sep 2005	Jul
8	7	6	5	4	3	May 2006	Mar 2006	Jan 2006	Nov 2005	Sep 2005	Jul

The above data are kind of mistake by UCI when preparing the data, the data is made more sense if PAY_6 means payment for month SEP 2005, and PAY_1 is for APR 2005.

5.13 CONVENIENCE VARIABLES

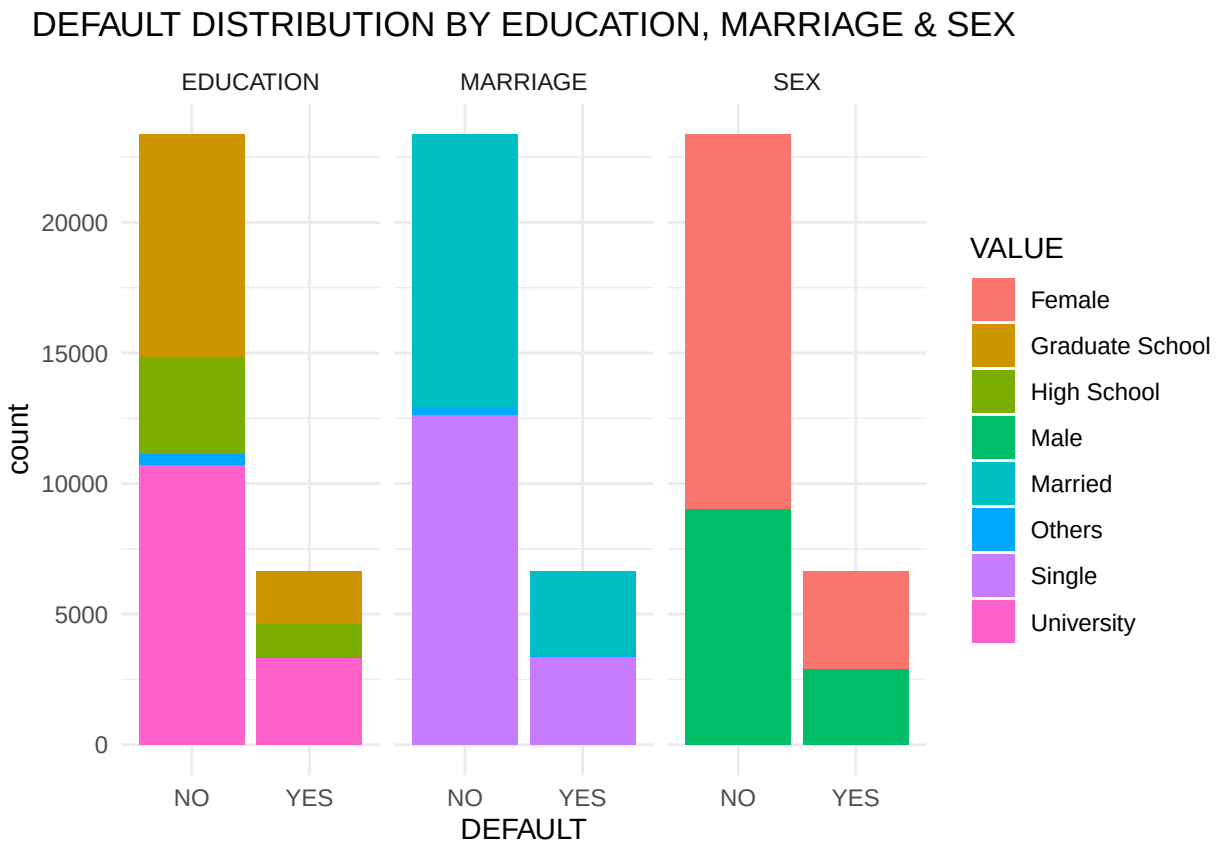
To simplified dataset presentation, I added following convenience variables to EDUCATION, SEX, MARRIAGE and DEFAULT VARIABLES:

- EDU_text with value ("Graduate School", "University", "High School" and "Others")
- SEX_text with value ("Male" and "Female")
- MAR_text with value ("Married", "Single" and "Others")
- DEF_text with value ("YES" and "NO")

```
dat_raw <- dat_raw %>% mutate(EDU_text = case_when(
  EDUCATION == 1 ~ "Graduate School",
  EDUCATION == 2 ~ "University",
  EDUCATION == 3 ~ "High School",
  TRUE ~ "Others"
)) %>% mutate(SEX_text=ifelse(SEX==1,"Male","Female")) %>%
  mutate(MAR_text = case_when(
    MARRIAGE == 1 ~ "Married",
    MARRIAGE == 2 ~ "Single",
    TRUE ~ "Others"
  )) %>% mutate(DEF_text=ifelse(DEFAULT==1,"YES","NO"))
```

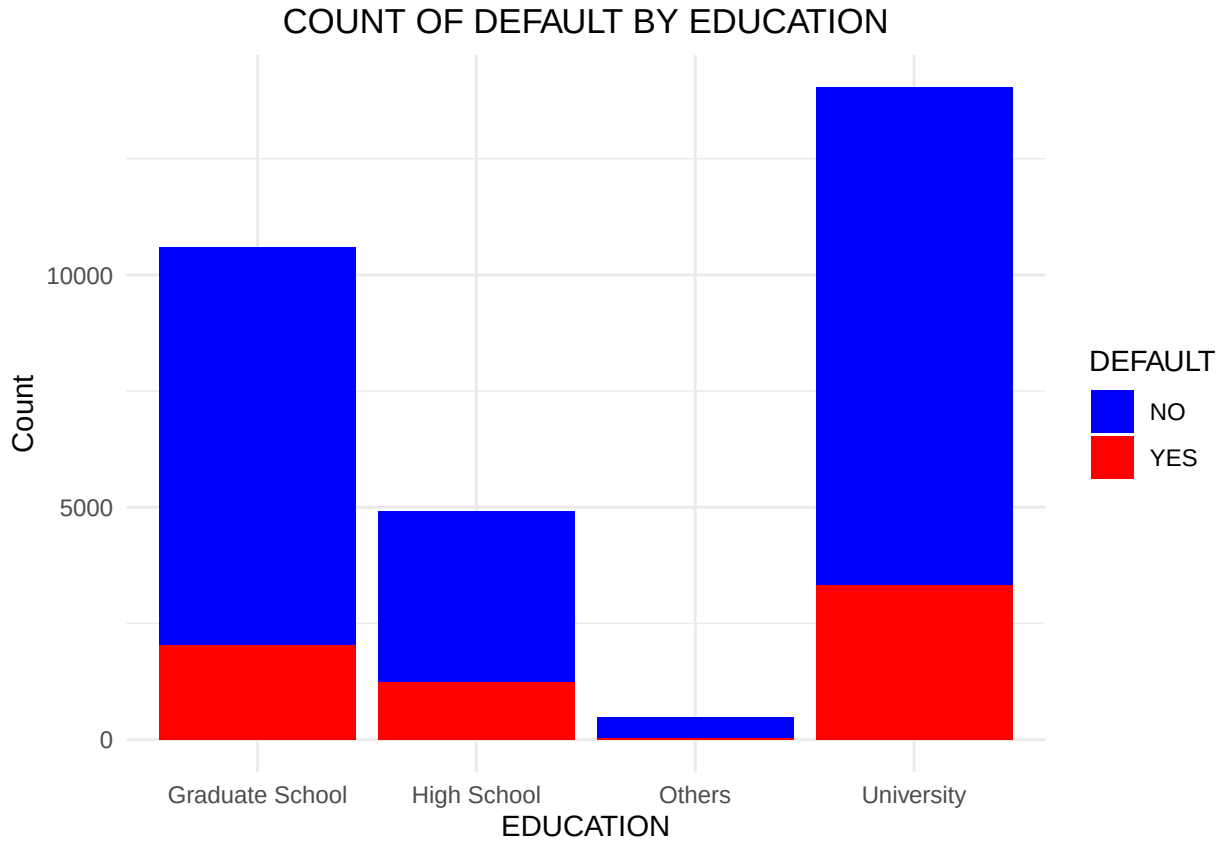
6 Explore Data

In this section I am exploring data from the perspective of how DEFAULT correlate with other variables. Below is the overall DEFAULT versus SEX, EDUCATION & MARRIAGE.

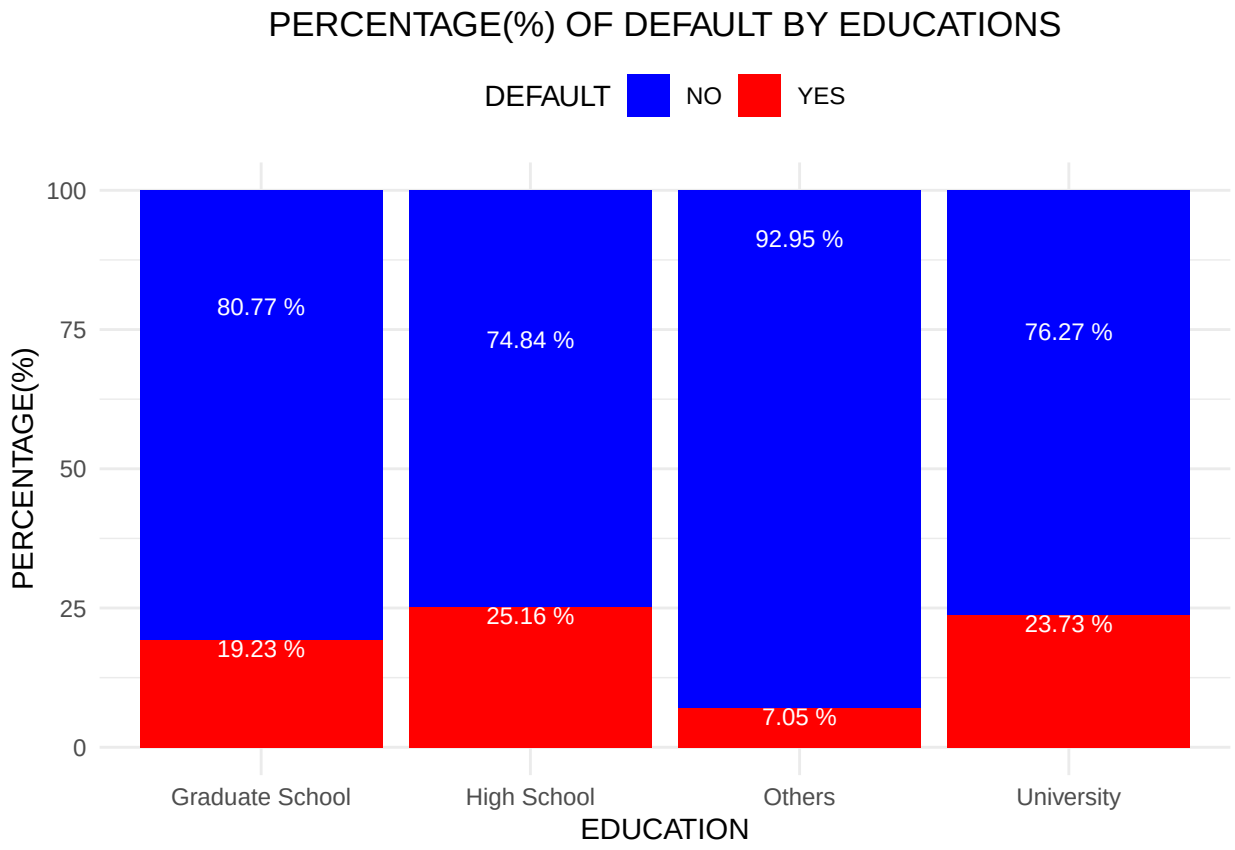


6.1 EDUCATION VS DEFAULT

From below chart, DEFAULT happens in all EDUCATION categories.

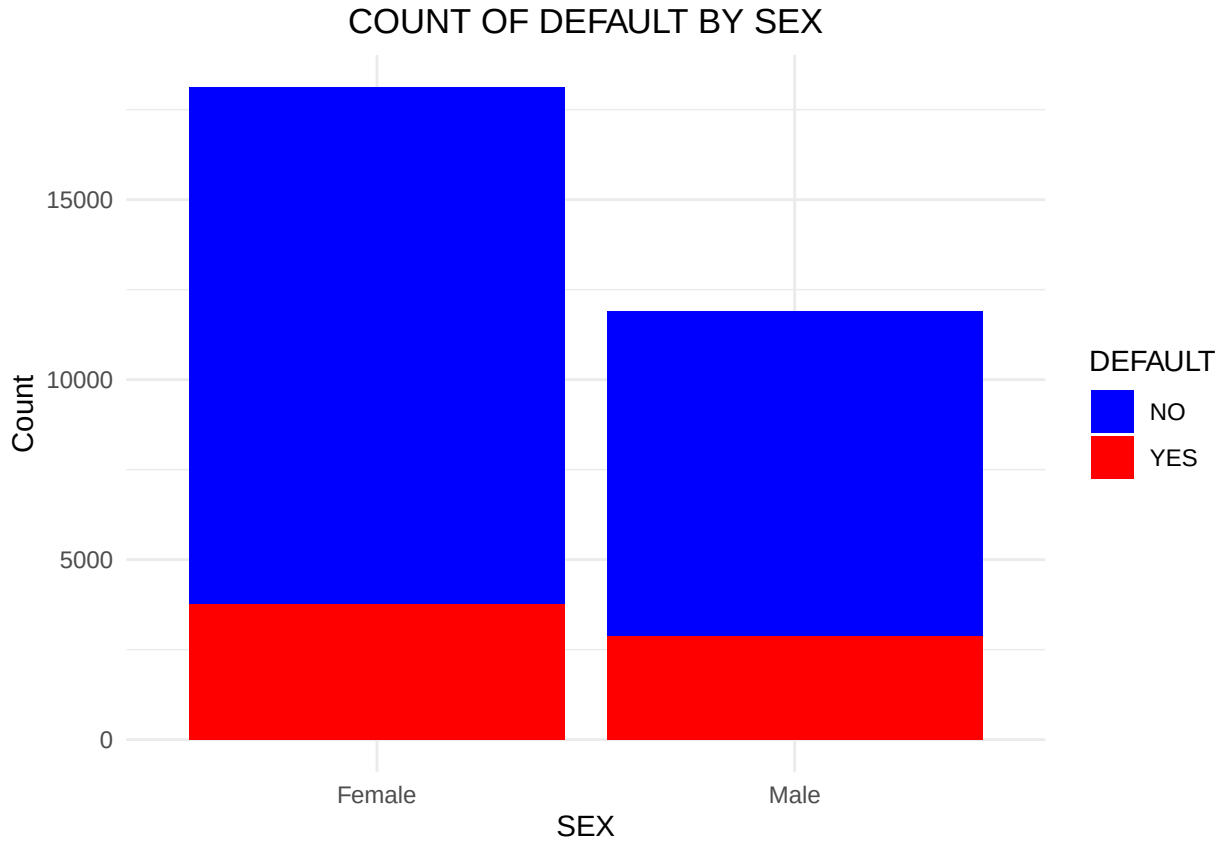


By changing the count into percentage, in exception of "Others", there is no obvious relationship between education level and likelihood of a client default on OCT 2005 payment.

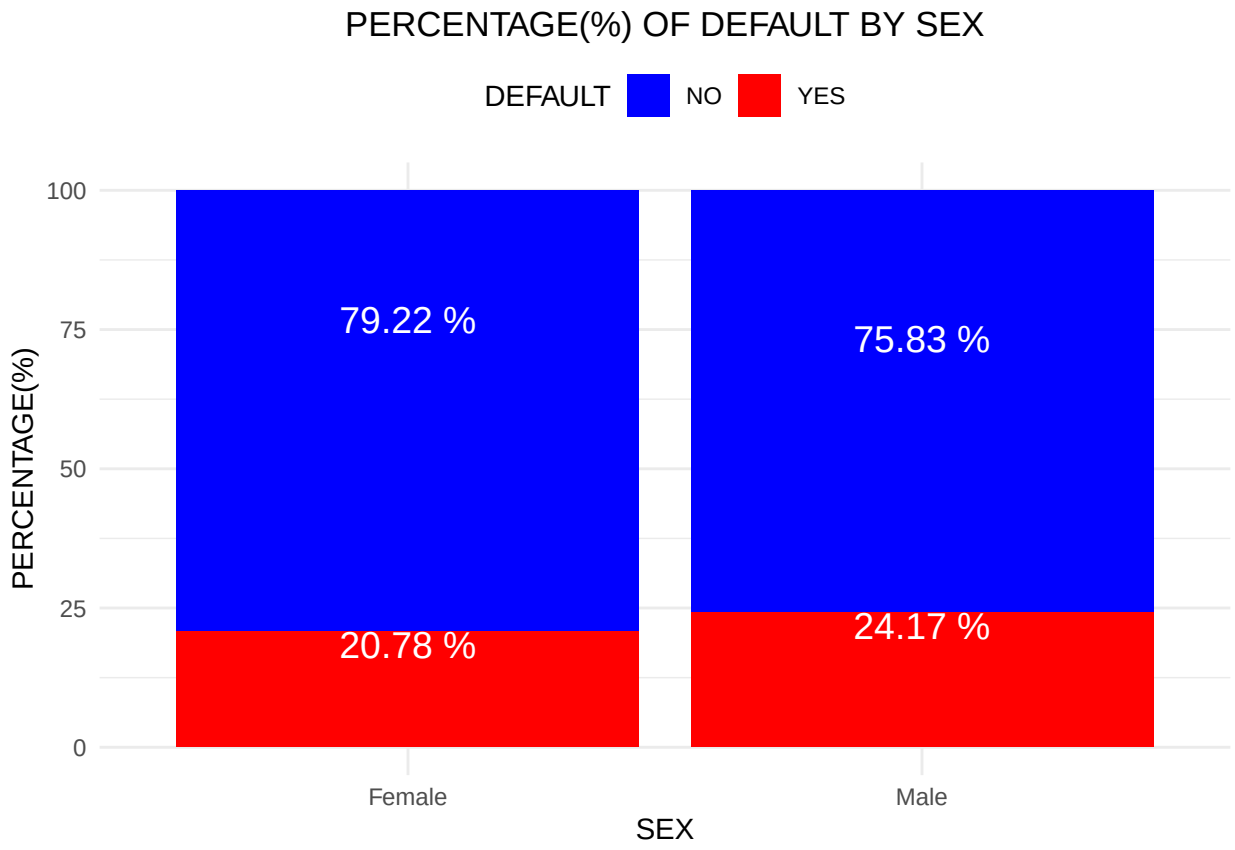


6.2 SEX vs DEFAULT

From the following chart, both sex had defaulted OCT 2005 payment.

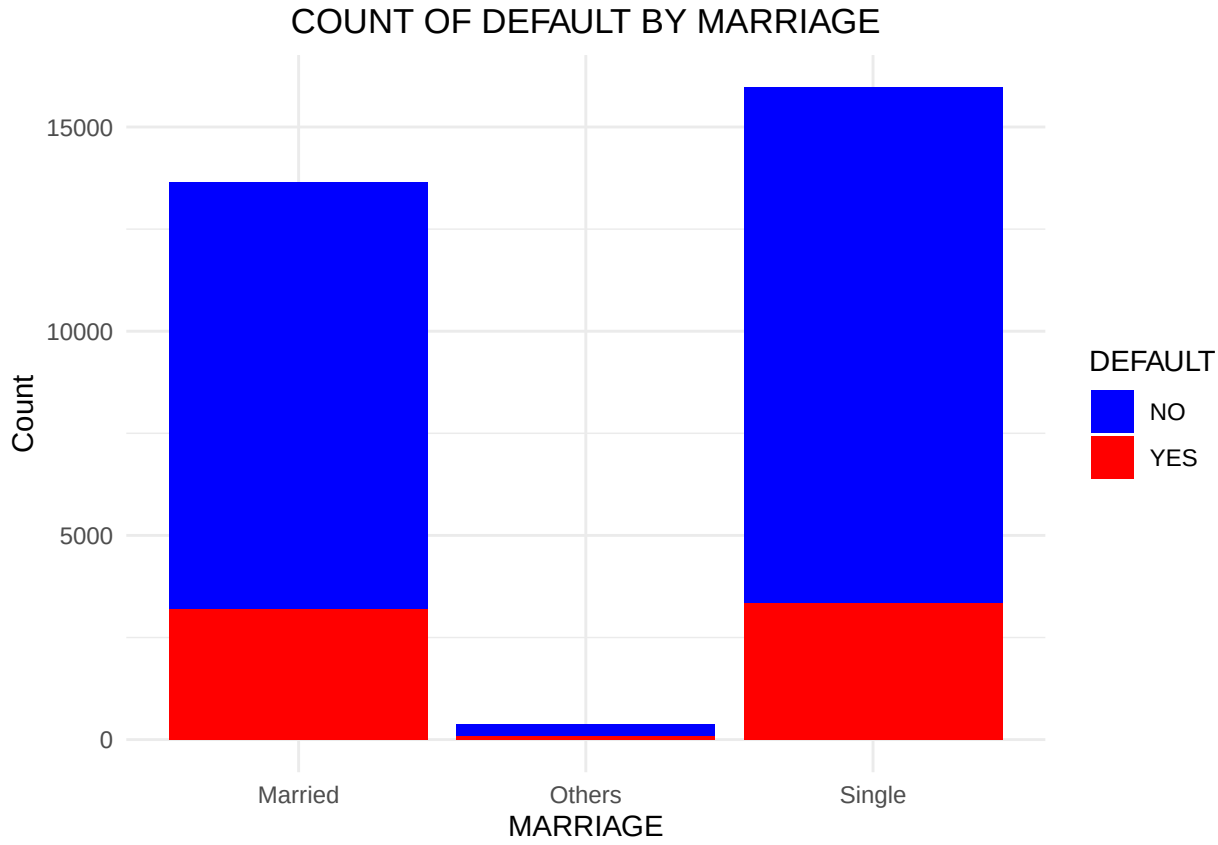


By analysing the DEFAULT percentage between SEX, there's no significantly difference between MALE & FEMALE on the likelihood of defaulting OCT 2005, however, with MALE has slightly higher by 0.1631%.

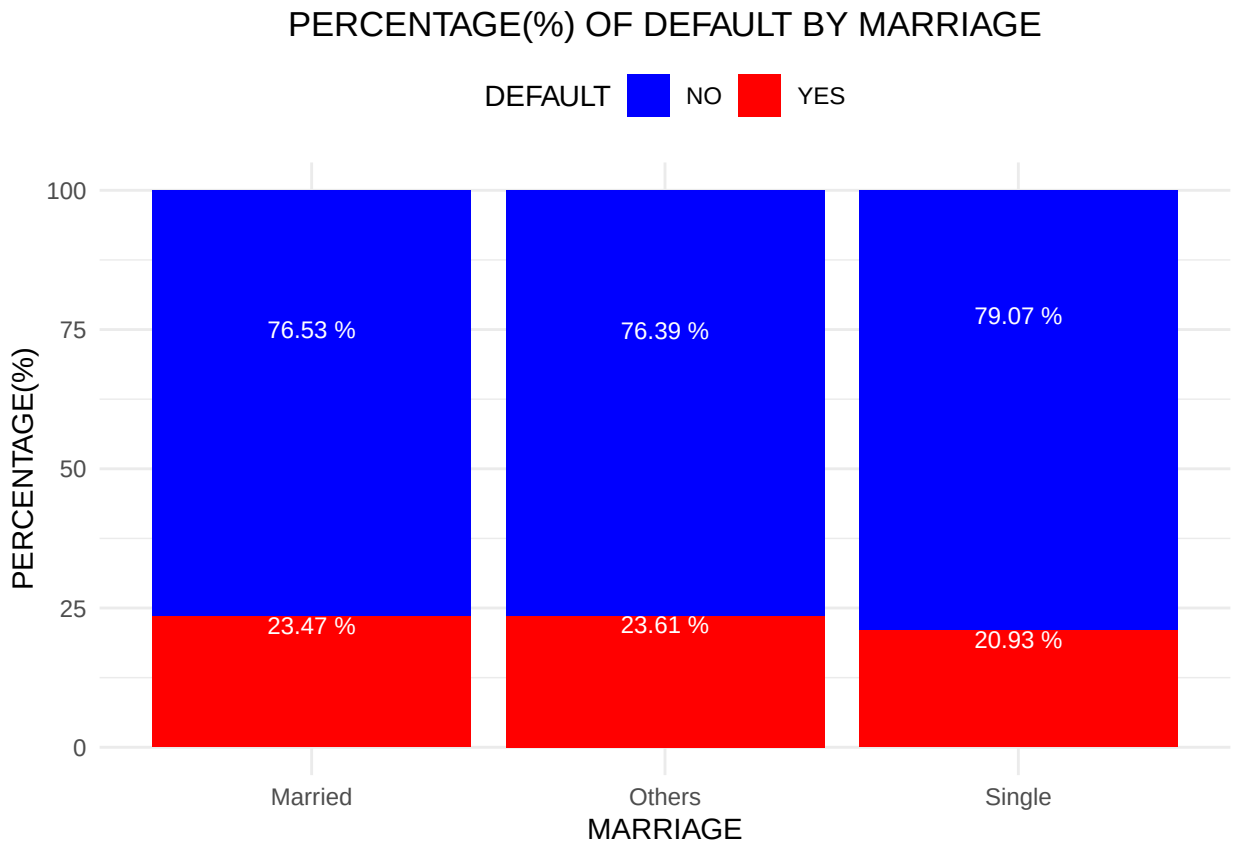


6.3 MARRIAGE vs DEFAULT

Following chart show count of DEFAULT by MARRIAGE.

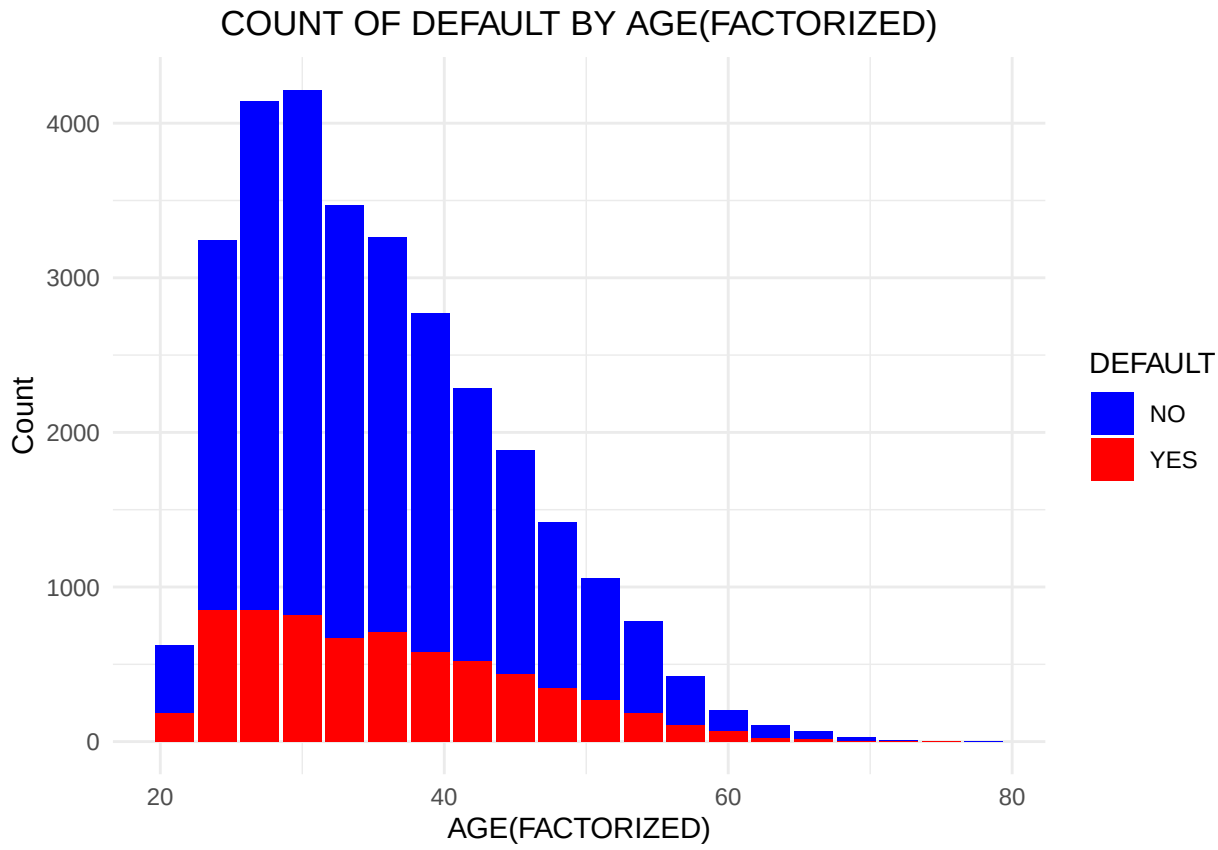


By plotting DEFAULT percentage between MARRIAGE, there is no apparant different between the possibility of defaulting OCT 2005 payment by Marriage status.

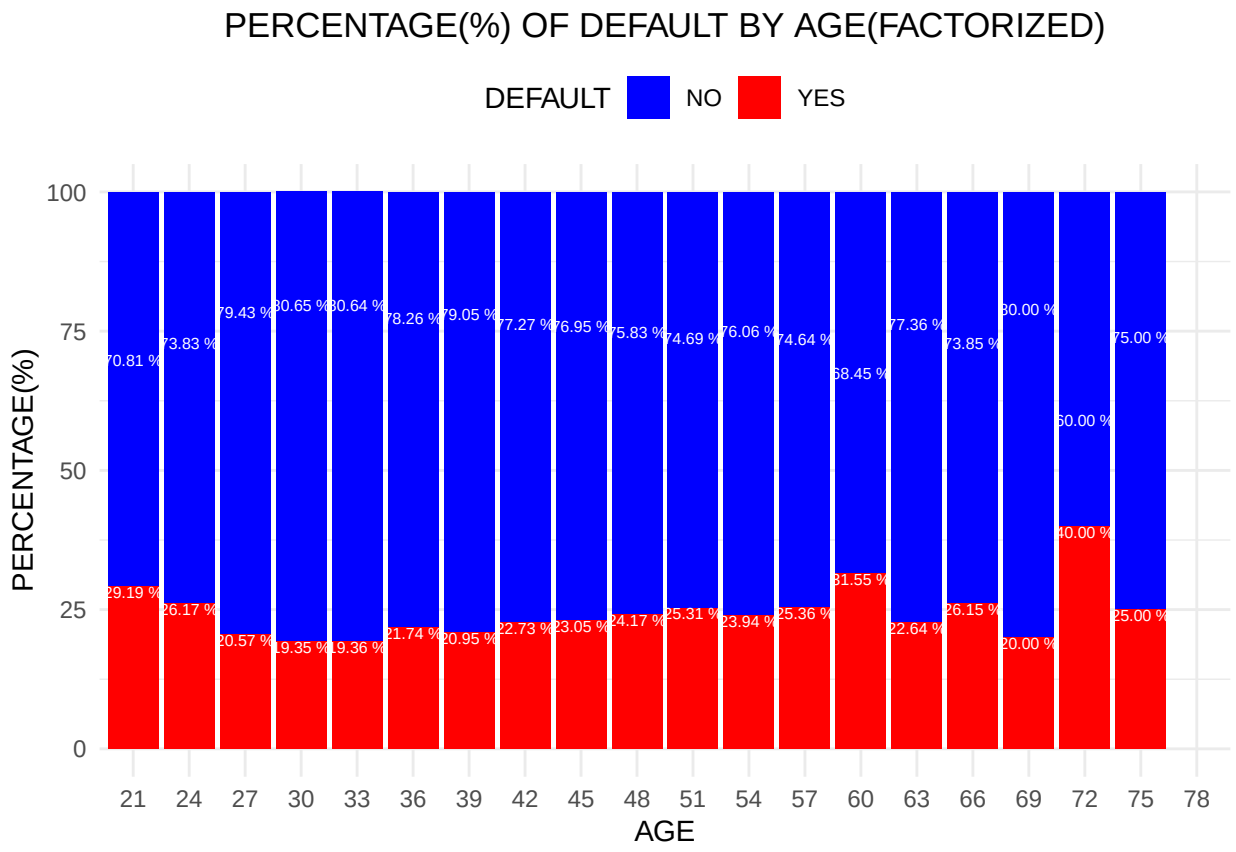


6.4 AGE_FAC vs DEFAULT

Following chart shows that default on OCT 2005 payment happens at clients of all ages.



Performing percentage of DEFAULT by AGE(Factorized) analysis, I notice that age group 72 (71 to 73) has the most high possibility of defaulting (40%) whilst those in age group 30 (29 to 31) has the lowest possibility of defaulting OCT 2005 credit card payment. The AGE_FAC has some impact to the possibility of DEFAULT.

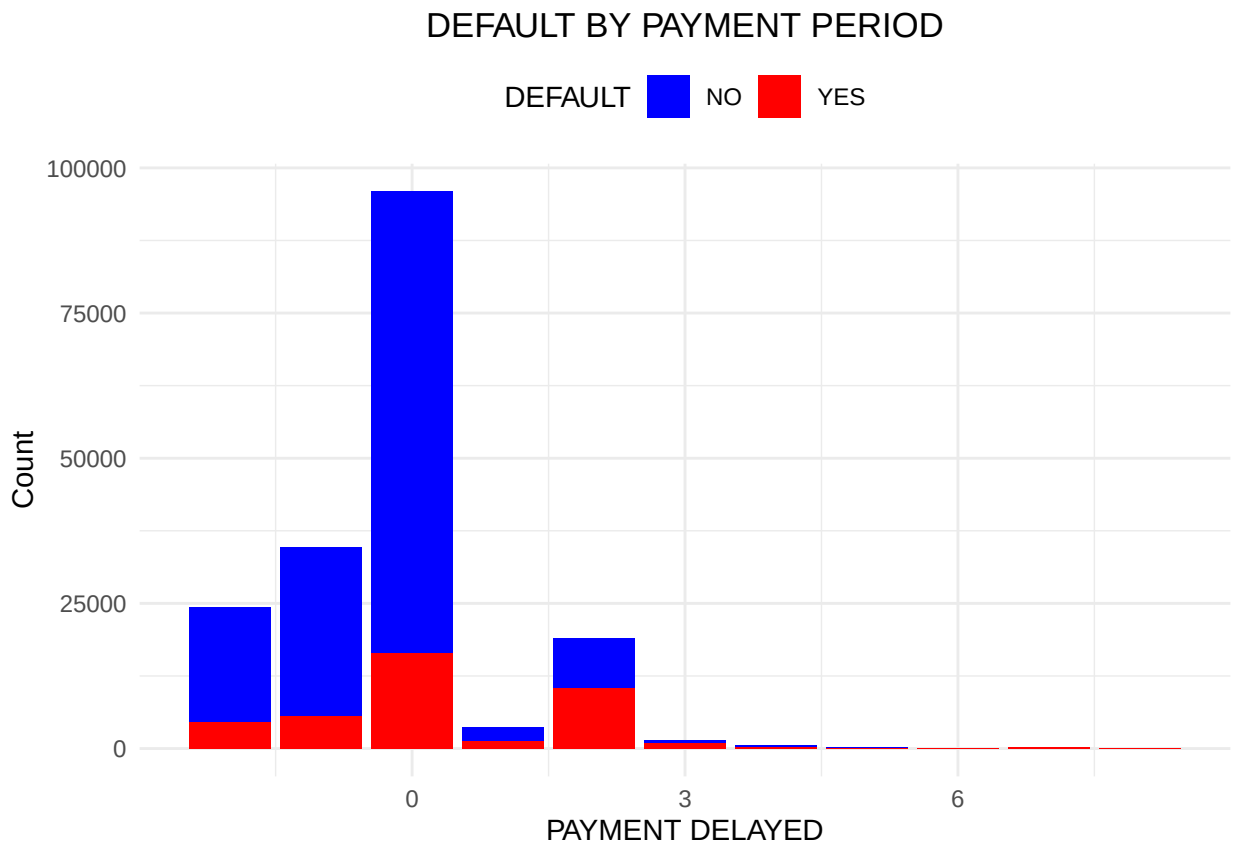


6.5 PAY_1 vs DEFAULT

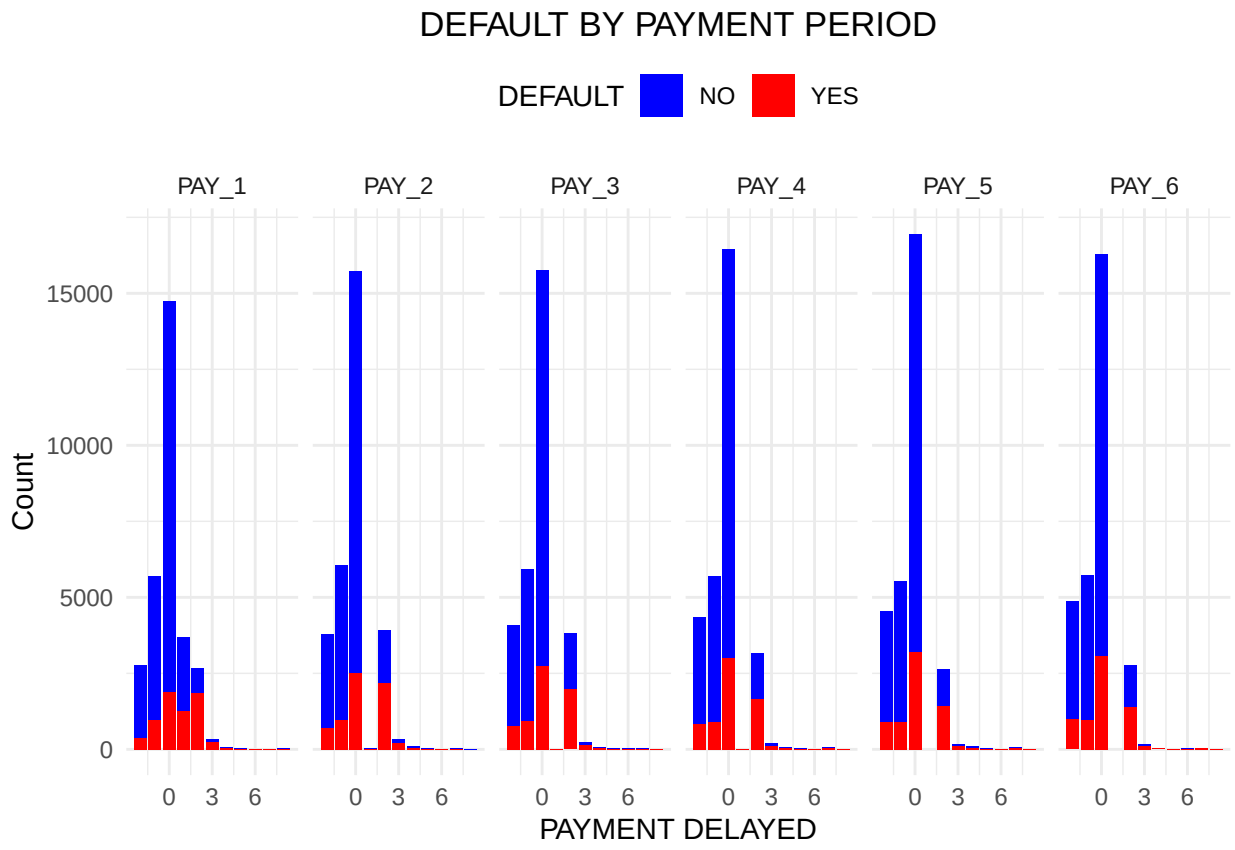
Under credit card industry definition, DEFAULT does not happens if the client paid at least the minimal amount before payment due date. This means, if a client default on Sep 2005, it does not means he will also default in Oct 2005, thus:

- PAY_1 >= 1 (default on SEP 2005) does not necessary means DEFAULT = 1 (default on OCT 2005), and
- PAY_1 <= -1 (pay on time for SEP 2005) does not necessary means DEFAULT = 0 (pay on time for OCT 2005).

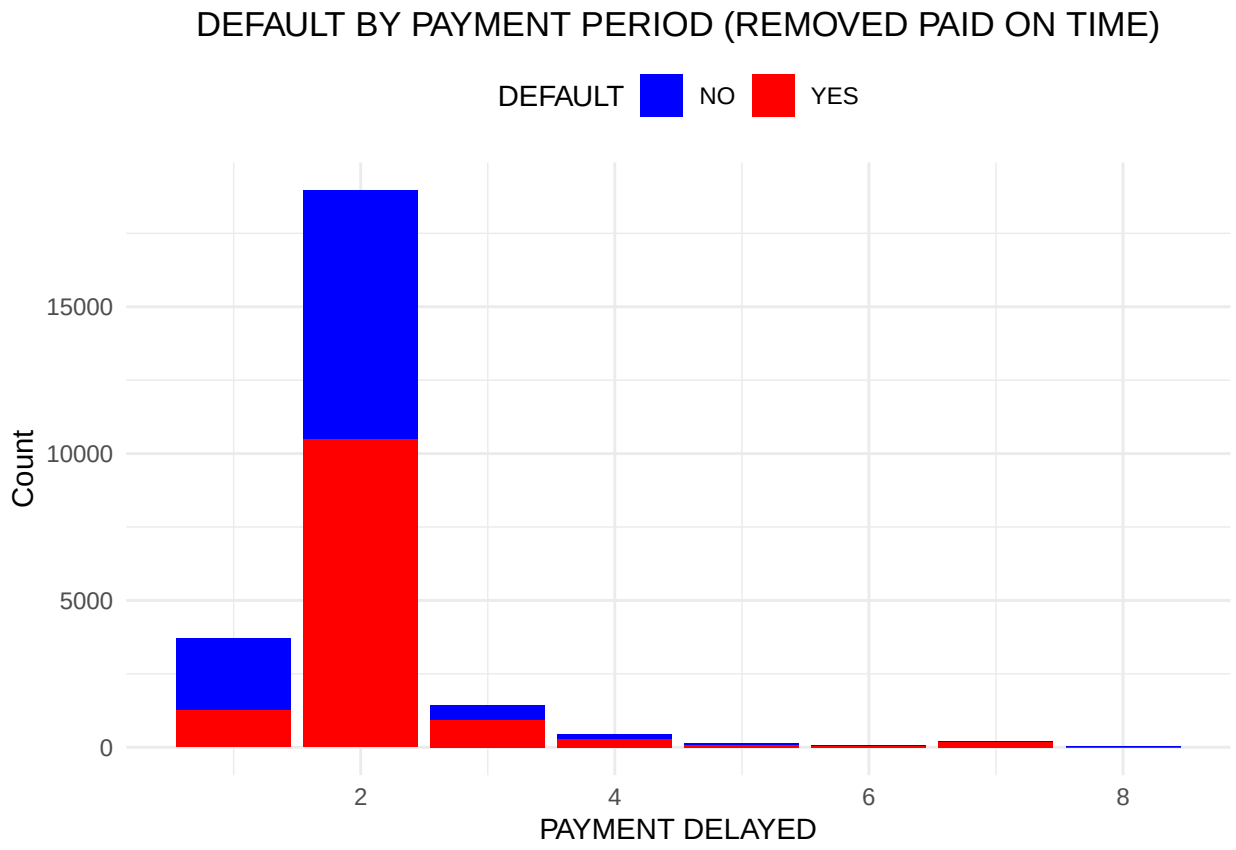
As following chart shown, a client who pay on time, may still default OCT 2005 payment:



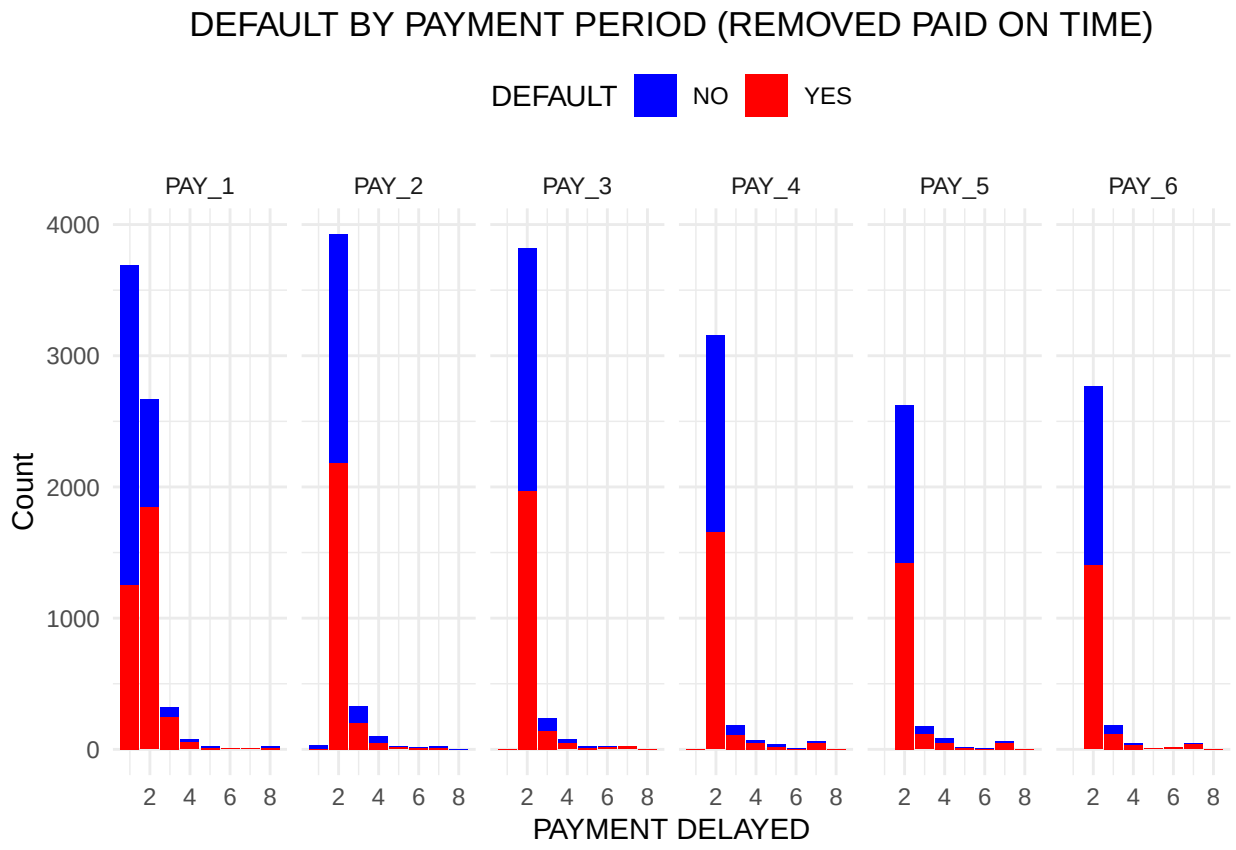
By plotting payment status for Apr 2005 (PAY_6) to Sep 2005 (PAY_1), there's no significant different between the probability of DEFAULT from the 6 months payment status history.



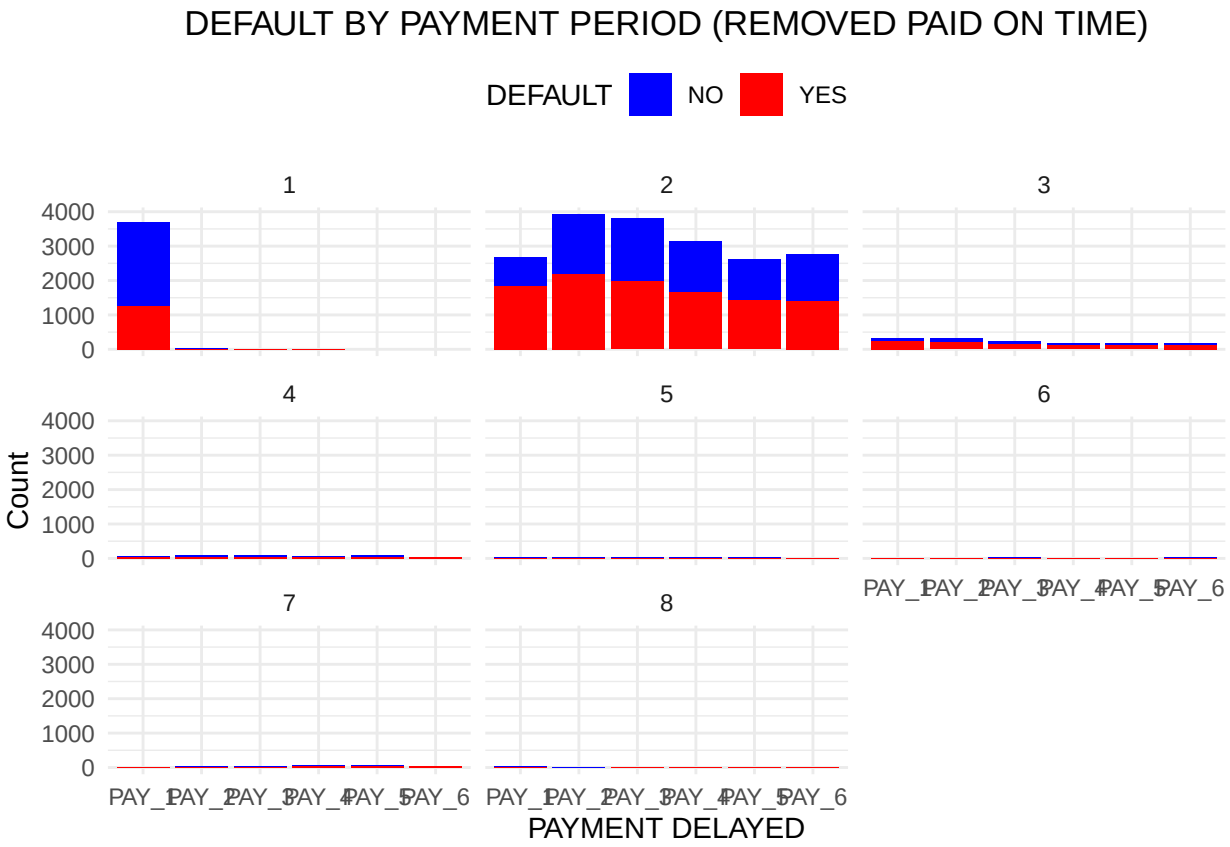
However, if I removed those month a client pay on time, there is obvios indication that a client will likely to DEFAULT OCT 2005, if he delayed payment for 2 months.



The observation become more clear when plotting monthly payment delay (PAY_1 ... PAY_6). A client will likely to default on Oct 2005 payment if he delayed payment for 2 months.



Following chart further support my hypothesis:

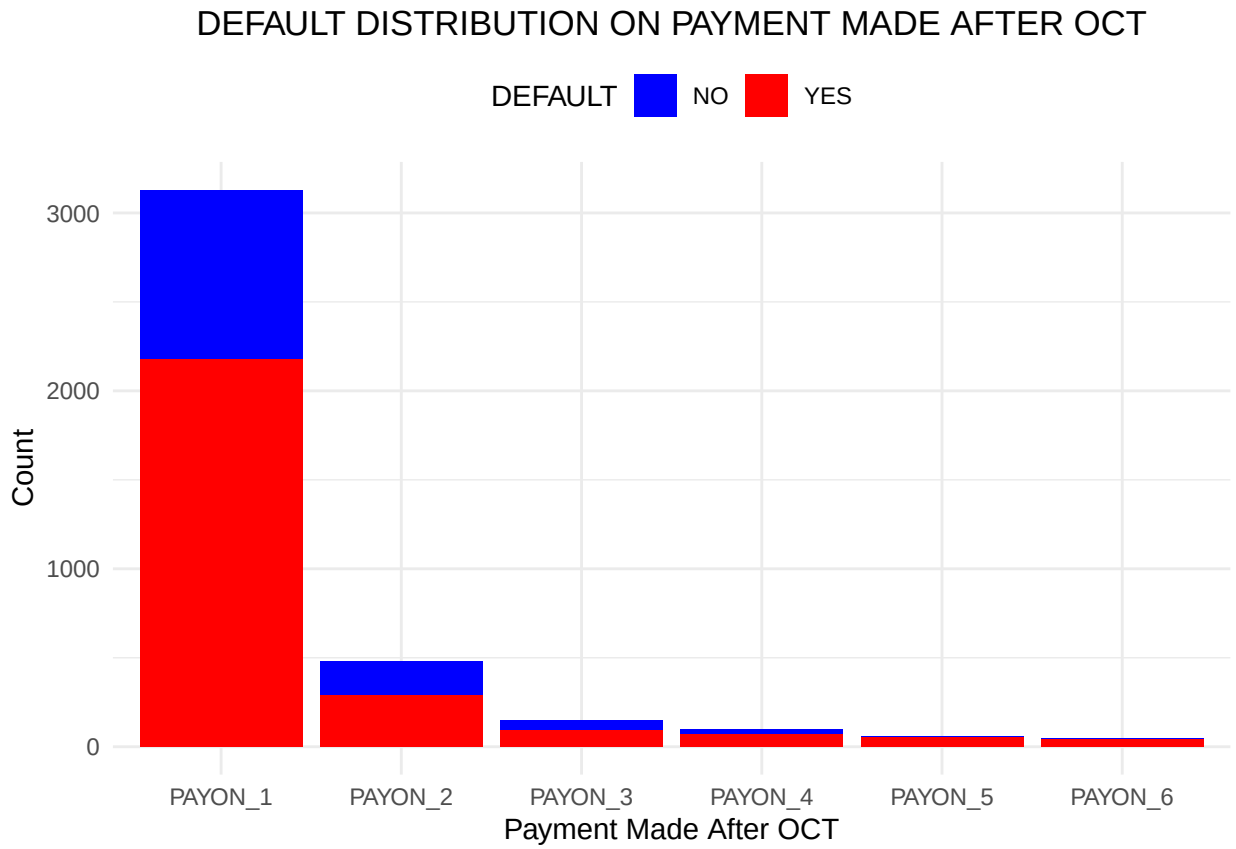


6.5 PAYON

The PAY_x above means the number of month of payment delayed. As my analysis is on DEFAULT payment for OCT 2005, I need to take a snopshot of payment status as of OCT 2005, and not after this. I introduce a new variable PAYON which means whether the payment has been done for the past months when the calendar month is OCT 2005.

```
dat_raw <- dat_raw %>% mutate(PAYON_1 = ifelse(PAY_1>1,1,0),
                              PAYON_2 = ifelse(PAY_2>2,1,0),
                              PAYON_3 = ifelse(PAY_3>3,1,0),
                              PAYON_4 = ifelse(PAY_4>4,1,0),
                              PAYON_5 = ifelse(PAY_5>5,1,0),
                              PAYON_6 = ifelse(PAY_6>6,1,0))
```

From following PAYON chart, it's very obvious that if a client still has not settle payment for SEP 2005 (PAYON_1 =1), he has a high probability of 69.55% of defaulting payment for OCT 2005 (DEFAULT = 1).

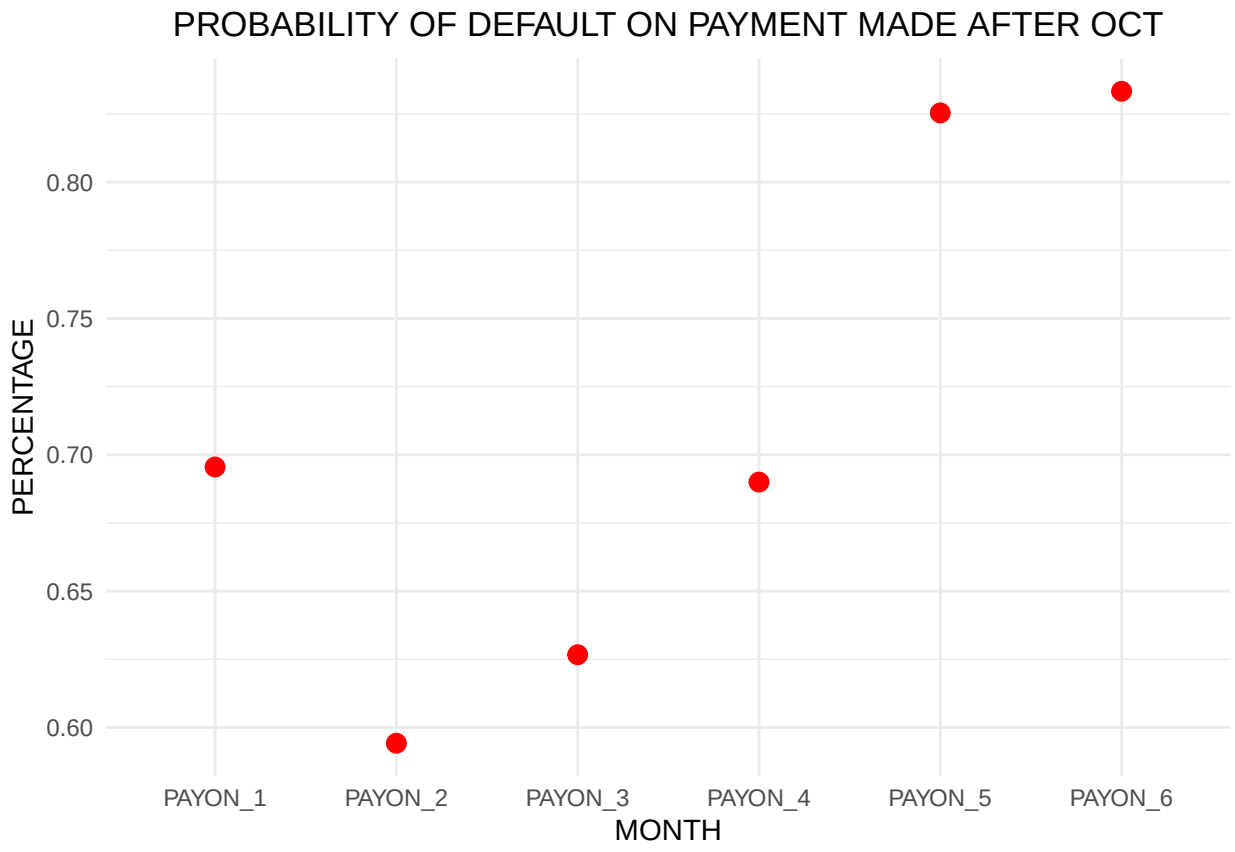


The observation also tells that the probability of defaulting OCT 2005 is highest if the client has yet to pay for SEP 2005 bill, compare with Aug 2005, July 2005 ... April 2005 bills, by a factor of 6.4803 times. This can be explained that the client has more time to pay for past bills compare with Sep 2005 bills.

The probability of default vs Pay on time for Oct is higher, if a client default on a month. Also, the default probability is higher, if client yet to pay for historical default, as illustrate in following table:

Table 10: PAYON SUMMARY

PAYON	NO	YES	MTH_DEFAULT	PROB_DEFAULT
PAYON_1	953	2,177	3,130	0.6955
PAYON_2	196	287	483	0.5942
PAYON_3	56	94	150	0.6267
PAYON_4	31	69	100	0.6900
PAYON_5	11	52	63	0.8254
PAYON_6	8	40	48	0.8333



6.6 PAYDUE

Based on above analysis of PAYON_x variables, here I introduce a new variable PAYDUE which is a derived from the probability of a client not paying OCT 2005 Credit Card Bill, PAYON_x. The sum of PAYON_x with weightage is use to value each PAYON contribution to overall probability of DEFAULT.

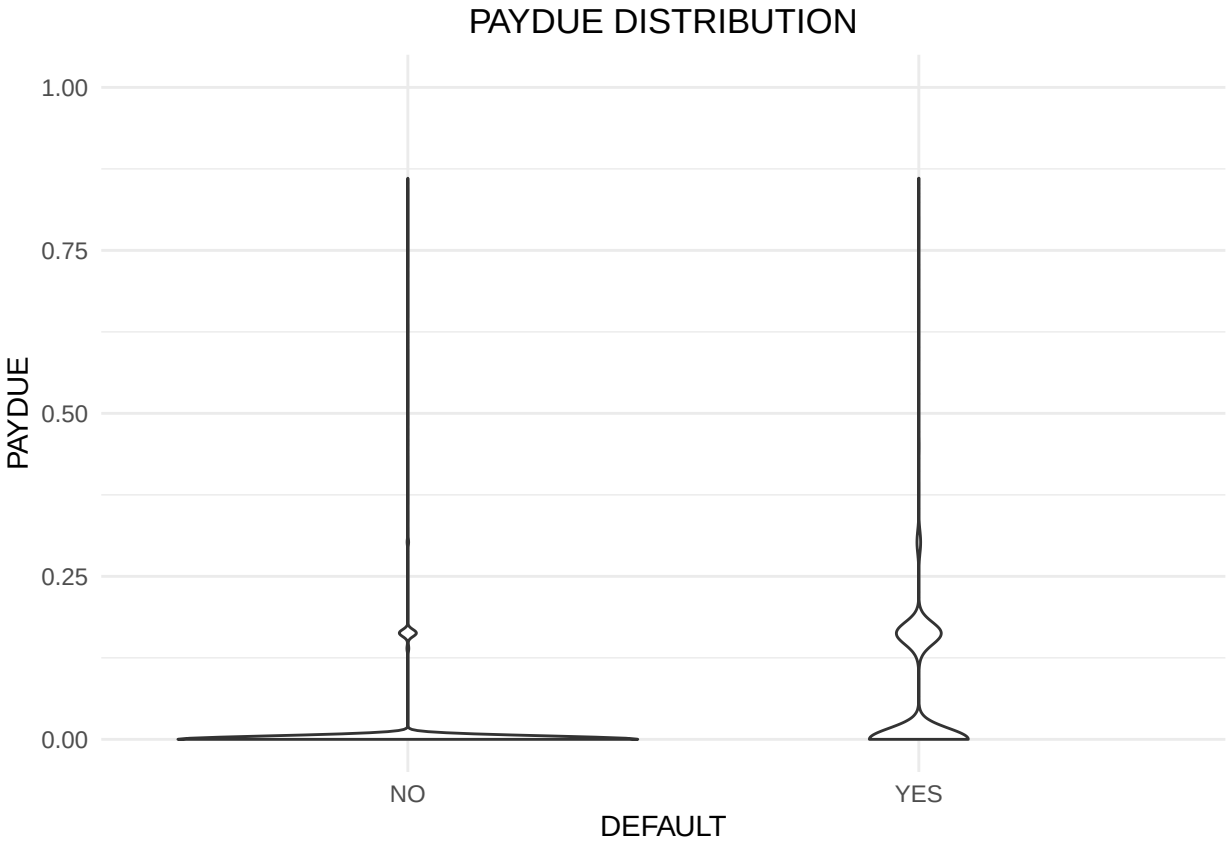
PAYDUE is derived from the following codes:

```
tmp_PAYDUE <- dat_payon_sum$PROB_DEFAULT/(sum(dat_payon_sum$PROB_DEFAULT))
dat_raw <- dat_raw %>% mutate(PAYDUE=(as.numeric(PAYON_1)*tmp_PAYDUE[1])+
                                (PAYON_2*tmp_PAYDUE[2])+
                                (PAYON_3*tmp_PAYDUE[3])+
                                (PAYON_4*tmp_PAYDUE[4])+
                                (PAYON_5*tmp_PAYDUE[5])+
                                (PAYON_6*tmp_PAYDUE[6]))
```

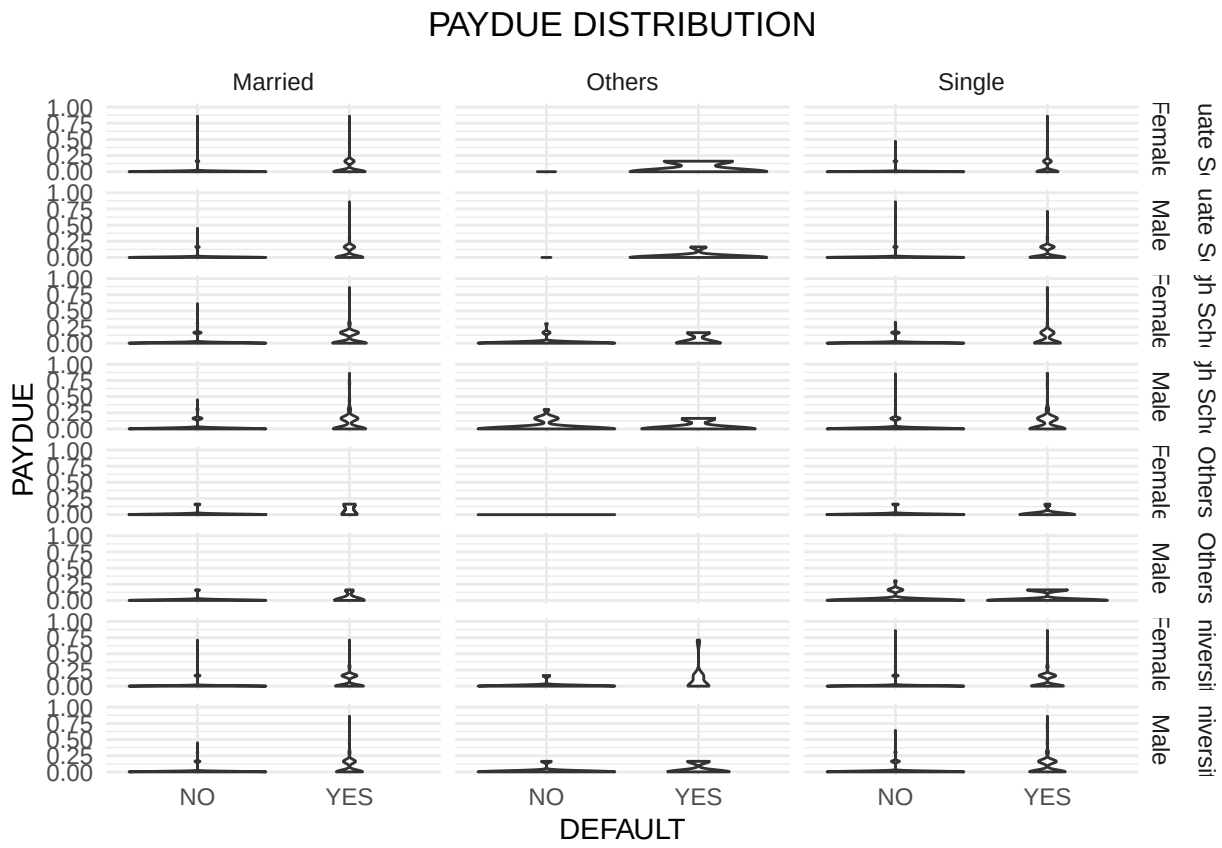
Table 11: SAMPLE PAYDUE

PAYDUE	PAYON_1	PAYON_2	PAYON_3	PAYON_4	PAYON_5	PAYON_6
0.7138	1	0	0	1	1	1
0.3024	1	1	0	0	0	0
0.5184	1	0	0	1	1	0
0.3024	1	1	0	0	0	0
0.3024	1	1	0	0	0	0
0.3100	1	0	1	0	0	0
0.3024	1	1	0	0	0	0
0.3024	1	1	0	0	0	0
0.7138	1	0	0	1	1	1
0.7138	1	0	0	1	1	1

From the following PAYDUE Distribution, the possibility of DEFAULT increase when probability of PAYDUE increase:



Following cross analysis between PAYDUE with CATEGORICAL VARIABLE EDUCATION, SEX & MARRIAGE status presented clear illustration on the relationship between PAYDUE and DEFAULT.



6.7 Correlation Analysis

I performed correlation analysis and principal components analysis on numerical fields, and as observed in previous sections, DEFAULT is associate with PAYDUE, and loosely with others (most with reverse effect).

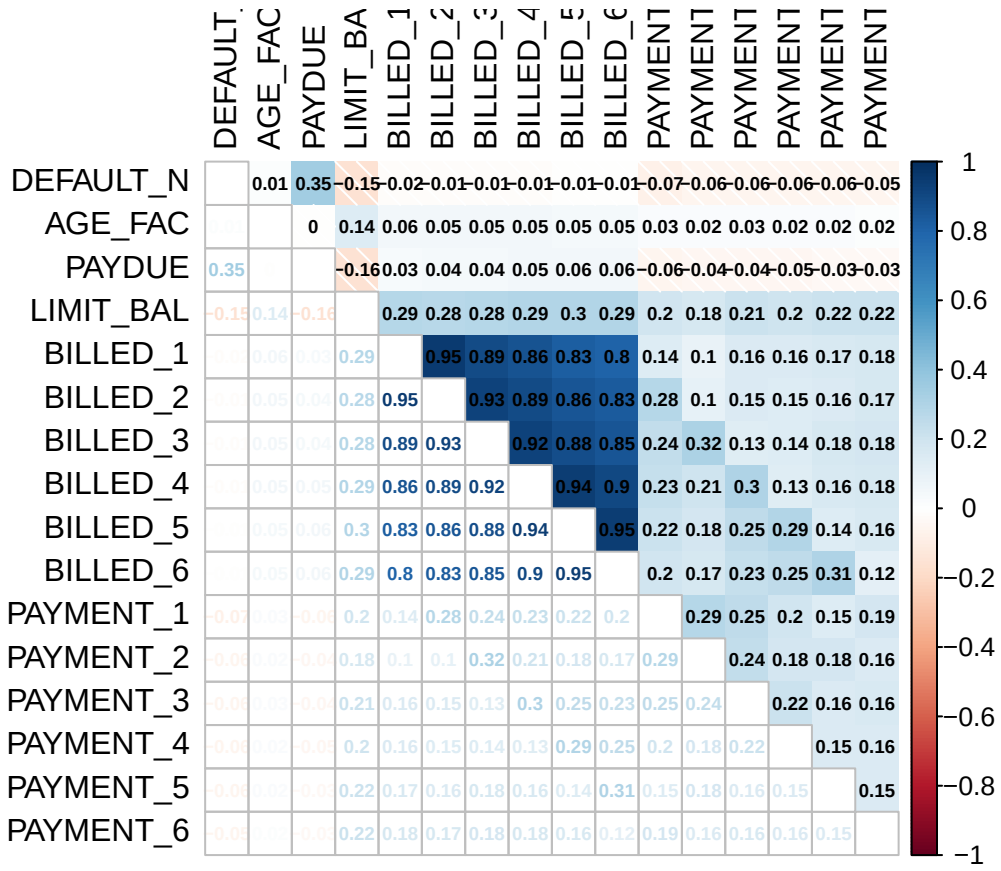


Table 12: DEFAULT CORRELATION

	x
DEFAULT_N	1.0000
AGE_FAC	0.0127
PAYDUE	0.3490
LIMIT_BAL	-0.1535
BILLED_1	-0.0196
BILLED_2	-0.0142
BILLED_3	-0.0141
BILLED_4	-0.0102
BILLED_5	-0.0068
BILLED_6	-0.0054
PAYMENT_1	-0.0729
PAYMENT_2	-0.0586
PAYMENT_3	-0.0563
PAYMENT_4	-0.0568
PAYMENT_5	-0.0551
PAYMENT_6	-0.0532

6.8 Revisit Variables Summary

Let's revisit dataset summary after above manipulations:

```
### Data Frame Summary
#### dat_raw
**Dimensions:** 30000 x 36
**Duplicates:** 35
```

No	Variable	Stats / Values	Freqs (% of Valid)
1	LIMIT_BAL\ [numeric]	Mean (sd) : 167484.3 (129747.7)\ min < med < max:\ 10000 < 140000 < 1000000\ IQR (CV) : 190000 (0.8)	81 distinct values
2	SEX\ [factor]	1\. 1\ 2\. 2	11888 (39.6%)\ 18112 (60.4%)
3	EDUCATION\ [factor]	1\. 0\ 2\. 1\ 3\. 2\ 4\. 3\ 5\. 4\ 6\. 5\ 7\. 6	0 (0.0%)\ 10585 (35.3%)\ 14030 (46.8%)\ 4917 (16.4%)\ 468 (1.6%)\ 0 (0.0%)\ 0 (0.0%)
4	MARRIAGE\ [factor]	1\. 0\ 2\. 1\ 3\. 2\ 4\. 3	0 (0.0%)\ 13659 (45.5%)\ 15964 (53.2%)\ 377 (1.3%)
5	AGE\ [integer]	Mean (sd) : 35.5 (9.2)\ min < med < max:\ 21 < 34 < 79\ IQR (CV) : 13 (0.3)	56 distinct values
6	PAY_1\ [integer]	Mean (sd) : 0 (1.1)\ min < med < max:\ -2 < 0 < 8\ IQR (CV) : 1 (-67.3)	11 distinct values
7	PAY_2\ [integer]	Mean (sd) : -0.1 (1.2)\ min < med < max:\ -2 < 0 < 8\ IQR (CV) : 1 (-8.9)	11 distinct values
8	PAY_3\ [integer]	Mean (sd) : -0.2 (1.2)\ min < med < max:\ -2 < 0 < 8\ IQR (CV) : 1 (-7.2)	11 distinct values
9	PAY_4\	Mean (sd) : -0.2 (1.2)\	11 distinct values

	[integer]	min < med < max:\ -2 < 0 < 8\ IQR (CV) : 1 (-5.3)		
10	PAY_5\ [integer]	Mean (sd) : -0.3 (1.1)\ min < med < max:\ -2 < 0 < 8\ IQR (CV) : 1 (-4.3)	-2 : 4546 (15.2%)\ -1 : 5539 (18.5%)\ 0 : 16947 (56.5%)\ 2 : 2626 (8.8%)\ 3 : 178 (0.6%)\ 4 : 84 (0.3%)\ 5 : 17 (0.1%)\ 6 : 4 (0.0%)\ 7 : 58 (0.2%)\ 8 : 1 (0.0%)	
11	PAY_6\ [integer]	Mean (sd) : -0.3 (1.1)\ min < med < max:\ -2 < 0 < 8\ IQR (CV) : 1 (-4)	-2 : 4895 (16.3%)\ -1 : 5740 (19.1%)\ 0 : 16286 (54.3%)\ 2 : 2766 (9.2%)\ 3 : 184 (0.6%)\ 4 : 49 (0.2%)\ 5 : 13 (0.0%)\ 6 : 19 (0.1%)\ 7 : 46 (0.1%)\ 8 : 2 (0.0%)	
12	BILLED_1\ [numeric]	Mean (sd) : 51223.3 (73635.9)\ min < med < max:\ -165580 < 22381.5 < 964511\ IQR (CV) : 63532.2 (1.4)	22723 distinct values	
13	BILLED_2\ [numeric]	Mean (sd) : 49179.1 (71173.8)\ min < med < max:\ -69777 < 21200 < 983931\ IQR (CV) : 61021.5 (1.4)	22346 distinct values	
14	BILLED_3\ [numeric]	Mean (sd) : 47013.2 (69349.4)\ min < med < max:\ -157264 < 20088.5 < 1664089\ IQR (CV) : 57498.5 (1.5)	22026 distinct values	
15	BILLED_4\ [numeric]	Mean (sd) : 43262.9 (64332.9)\ min < med < max:\ -170000 < 19052 < 891586\ IQR (CV) : 52179.2 (1.5)	21548 distinct values	
16	BILLED_5\ [numeric]	Mean (sd) : 40311.4 (60797.2)\ min < med < max:\ -81334 < 18104.5 < 927171\ IQR (CV) : 48427.5 (1.5)	21010 distinct values	
17	BILLED_6\ [numeric]	Mean (sd) : 38871.8 (59554.1)\ min < med < max:\ -339603 < 17071 < 961664\	20604 distinct values	

		IQR (CV) : 47942.2 (1.5)	
18	PAYMENT_1\ [numeric]	Mean (sd) : 5663.6 (16563.3)\ min < med < max:\ 0 < 2100 < 873552\ IQR (CV) : 4006 (2.9)	7943 distinct values
19	PAYMENT_2\ [numeric]	Mean (sd) : 5921.2 (23040.9)\ min < med < max:\ 0 < 2009 < 1684259\ IQR (CV) : 4167 (3.9)	7899 distinct values
20	PAYMENT_3\ [numeric]	Mean (sd) : 5225.7 (17607)\ min < med < max:\ 0 < 1800 < 896040\ IQR (CV) : 4115 (3.4)	7518 distinct values
21	PAYMENT_4\ [numeric]	Mean (sd) : 4826.1 (15666.2)\ min < med < max:\ 0 < 1500 < 621000\ IQR (CV) : 3717.2 (3.2)	6937 distinct values
22	PAYMENT_5\ [numeric]	Mean (sd) : 4799.4 (15278.3)\ min < med < max:\ 0 < 1500 < 426529\ IQR (CV) : 3779 (3.2)	6897 distinct values
23	PAYMENT_6\ [numeric]	Mean (sd) : 5215.5 (17777.5)\ min < med < max:\ 0 < 1500 < 528666\ IQR (CV) : 3882.2 (3.4)	6939 distinct values
24	DEFAULT\ [factor]	1\. 0\ 2\. 1	23364 (77.9%)\ 6636 (22.1%)
25	AGE_FAC\ [numeric]	Mean (sd) : 35.5 (9.3)\ min < med < max:\ 21 < 33 < 78\ IQR (CV) : 15 (0.3)	20 distinct values
26	EDU_text\ [character]	1\. Graduate School\ 2\. High School\ 3\. Others\ 4\. University	10585 (35.3%)\ 4917 (16.4%)\ 468 (1.6%)\ 14030 (46.8%)
27	SEX_text\ [character]	1\. Female\ 2\. Male	18112 (60.4%)\ 11888 (39.6%)
28	MAR_text\ [character]	1\. Married\ 2\. Others\ 3\. Single	13659 (45.5%)\ 377 (1.3%)\ 15964 (53.2%)
29	DEF_text\ [character]	1\. NO\ 2\. YES	23364 (77.9%)\ 6636 (22.1%)

30	PAYON_1\ [numeric]	Min : 0\ Mean : 0.1\ Max : 1	0 : 26870 (89.6%)\ 1 : 3130 (10.4%)
31	PAYON_2\ [numeric]	Min : 0\ Mean : 0\ Max : 1	0 : 29517 (98.4%)\ 1 : 483 (1.6%)
32	PAYON_3\ [numeric]	Min : 0\ Mean : 0\ Max : 1	0 : 29850 (99.5%)\ 1 : 150 (0.5%)
33	PAYON_4\ [numeric]	Min : 0\ Mean : 0\ Max : 1	0 : 29900 (99.7%)\ 1 : 100 (0.3%)
34	PAYON_5\ [numeric]	Min : 0\ Mean : 0\ Max : 1	0 : 29937 (99.8%)\ 1 : 63 (0.2%)
35	PAYON_6\ [numeric]	Min : 0\ Mean : 0\ Max : 1	0 : 29952 (99.8%)\ 1 : 48 (0.2%)
36	PAYDUE\ [numeric]	Mean (sd) : 0 (0.1)\ min < med < max:\ 0 < 0 < 0.9\ IQR (CV) : 0 (3.2)	24 distinct values

7 Modeling

7.1 Training & Testing Dataset

My modeling begins with the separation of dataset into training dataset and testing dataset. The modeling will be performed on the training dataset, and evaluate using test dataset. Following script is used for the preparation of training dataset (dat_train, 80% of data) and testing dataset (dat_test, 20% of data):

```
set.seed(1, sample.kind = "Rounding")
tmp_idx_test <- createDataPartition(dat_raw$DEFAULT,times=1,p=0.2,list=FALSE)
dat_test <- dat_raw[tmp_idx_test,]
dat_train <- dat_raw[-tmp_idx_test,]

rm(tmp_idx_test)
```

Number of rows of each dataset are:

- training dataset (23,999 rows)
- testing dataset (6,001 rows)

7.2 Variables

Selection of variables for modeling are significant, balancing variable with interest and modeling performance. Based on correlation analysis, many variables are positive correlated, as such after careful evaluation and sampling, my modeling are performing with following variables:

“DEFAULT” “LIMIT_BAL” “SEX” “EDUCATION” “MARRIAGE” “AGE_FAC” “PAYDUE” “PAYON_1” “PAYON_2” “PAYON_3” “PAYON_4” “PAYON_5” “PAYON_6”

A new training dataset (dat_train_simple) & testing dataset (dat_test_simple) is prepared with only above columns.

```
dat_train_simple <- dat_train %>%
  select(DEFAULT, LIMIT_BAL, SEX, EDUCATION, MARRIAGE, AGE_FAC, PAYDUE,
         PAYON_1, PAYON_2, PAYON_3, PAYON_4, PAYON_5, PAYON_6)

dat_test_simple <- dat_test %>%
  select(DEFAULT, LIMIT_BAL, SEX, EDUCATION, MARRIAGE, AGE_FAC, PAYDUE,
         PAYON_1, PAYON_2, PAYON_3, PAYON_4, PAYON_5, PAYON_6)
```

7.3 Result Tibble

A result tibble is first defined to store the modeling result:

```
result <- tibble(Method = "",
                 Accuracy = 0,
                 Balanced = 0,
                 Sensitivity = 0,
                 Specificity = 0,
                 F1 = 0)
```

7.4 Generalized Linear Model

Modeling using Generalized Linear Model (GLM) method & the modeling result is as below:

```
glm_fit <- train(data=dat_train_simple, DEFAULT~.,method = "glm")
glm_predicted <- predict(glm_fit, newdata = dat_test)
glm_cm <- confusionMatrix(factor(glm_predicted),factor(dat_test$DEFAULT))
```

Table 13: PREDICTION RESULT USING GLM

	x
Method	GLM
Accuracy	0.825362439593401
Balanced Accuracy	0.649082043990213
Sensitivity	0.965332762679221
Specificity	0.332831325301205
F1	0.895928500496524

Following is the predicted Confusion Matrix:

```
                      Reference
Prediction             Actual Positive(1) Actual Negative(0)
  Predicted Positive(1)               4511                886
  Predicted Negative(0)                162                442
```

7.5 Linear Model

Modeling using Linear Model (LM) method & the modeling result is as below:

```
lm_fit <- lm(DEFAULT~.,
             data=dat_train_simple)
lm_predict <- predict(lm_fit, dat_test)
lm_predicted <- ifelse(lm_predict>1.5,1,0 )
lm_cm <- confusionMatrix(factor(lm_predicted),factor(dat_test$DEFAULT))
```

Table 14: PREDICTION RESULT USING LM

	x
Method	LM
Accuracy	0.825029161806366
Balanced Accuracy	0.650215590588332
Sensitivity	0.963834795634496
Specificity	0.336596385542169
F1	0.895605488168622

Following is the predicted Confusion Matrix:

```
                      Reference
Prediction             Actual Positive(1) Actual Negative(0)
  Predicted Positive(1)               4504                881
  Predicted Negative(0)                169                447
```

7.6 Generalized Additive Model using LOESS

Modeling using Generalized Additive Model (LOESS) method & the modeling result is as below:

```
loess_fit <- train(data=dat_train_simple, DEFAULT~., method = "gamLoess")
loess_predicted <- predict(loess_fit, dat_test)
loess_cm <- confusionMatrix(factor(loess_predicted),factor(dat_test$DEFAULT))
```

Table 15: PREDICTION RESULT USING GAMLOESS

	x
Method	LOESS
Accuracy	0.824195967338777
Balanced Accuracy	0.649950110736118
Sensitivity	0.962550823881875
Specificity	0.337349397590361
F1	0.89503531986867

Following is the predicted Confusion Matrix:

```
                      Reference
Prediction             Actual Positive(1) Actual Negative(0)
  Predicted Positive(1)               4498                880
  Predicted Negative(0)                175                448
```

7.7 K-nearest Neighbors

Modeling using K-nearest Neighbors method & the modeling result is as below:

```
set.seed(1,sample.kind = "Rounding")
knn_control <- trainControl(method = "repeatedcv", number = 10, p = .9, repeats=10)
knn_fit <- train(data=dat_train_simple, DEFAULT~., method = "knn",
                 tuneGrid = data.frame(k = seq(3, 21, by=2)),
                 trControl=knn_control)
knn_predicted <- predict(knn_fit, dat_test)
knn_cm <- confusionMatrix(factor(knn_predicted),factor(dat_test$DEFAULT))
```

Table 16: PREDICTION RESULT USING K-NN

	x
Method	K-NN
Accuracy	0.787202132977837
Balanced Accuracy	0.535641818289636
Sensitivity	0.986946287181682
Specificity	0.0843373493975904
F1	0.878392534044377

Following is the predicted Confusion Matrix:

```
                      Reference
Prediction             Actual Positive(1) Actual Negative(0)
  Predicted Positive(1)               4612                1216
  Predicted Negative(0)                 61                 112
```

7.8 K-Means Clustering

Modeling using K-Means Clustering method & the modeling result is as below:

```
set.seed(1, sample.kind = "Rounding")
f_predictkmeans <- function(x, k) {
  centers <- k$centers    # extract cluster centers
  # calculate distance to cluster centers
  distances <- sapply(1:nrow(x), function(i){
    apply(centers, 1, function(y) dist(rbind(x[i,], y)))
  })
  max.col(-t(distances))  # select cluster with min distance to center
}
k_means <- kmeans(dat_train_simple[,-1],centers=2)
k_predict <- f_predictkmeans(dat_test,k_means)
k_predicted <- ifelse(k_predict>1.5,0,1)
k_cm <- confusionMatrix(factor(k_predicted),factor(dat_test$DEFAULT))
```

Table 17: PREDICTION RESULT USING K-Means

	x
Method	K-Means
Accuracy	0.241959673387769
Balanced Accuracy	0.505182456769084
Sensitivity	0.0329552749839504
Specificity	0.977409638554217
F1	0.0634136298126416

Following is the predicted Confusion Matrix:

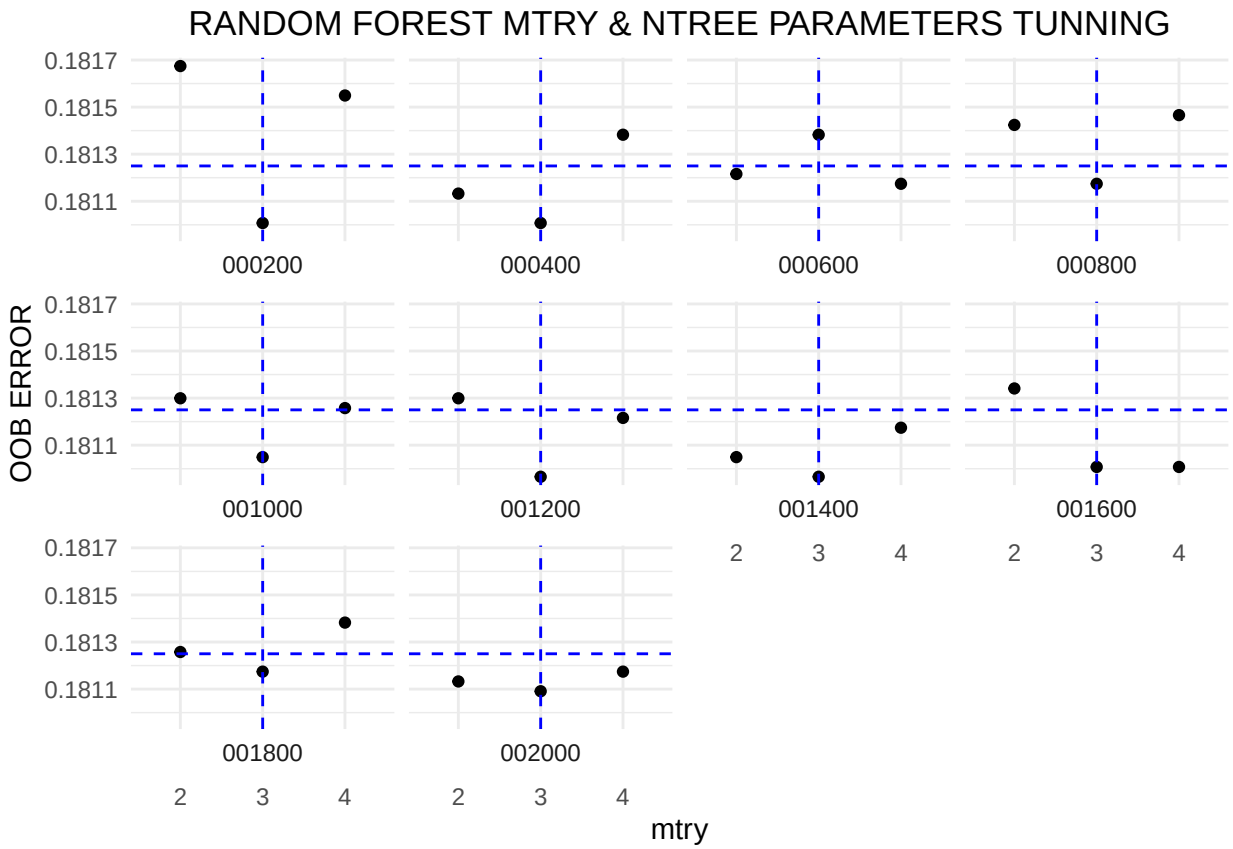
```
                     Reference
Prediction            Actual Positive(1) Actual Negative(0)
  Predicted Positive(1)                154                  30
  Predicted Negative(0)               4519                1298
```

7.9 Random Forest

Before performing using Random Forest method, I perform estimation to get the ntree & mtry parameters. Search for the optimal value (with respect to Out-of-Bag error estimate) of mtry:

```
set.seed(1,sample.kind = "Rounding")

rf_ntrees <- seq(200,by=200, len=10)
rf_tuning <- sapply(rf_ntrees, function(nt){
  t_mtry <- tuneRF(dat_train_simple[,-1],
                   dat_train_simple$DEFAULT,
                   stepFactor=1.5, improve=1e-5, ntree=nt,
                   plot=FALSE,trace=FALSE)
  t_mtry
})
```

```
-0.003683 0.00001
-0.002993 0.00001
-0.0006906 0.00001
-0.002072 0.00001
0.0009189 0.00001
0.0002299 0.00001
-0.01472 0.00001
-0.00138 0.00001
-0.00161 0.00001
-0.001381 0.00001
-0.001151 0.00001
-0.001842 0.00001
-0.001382 0.00001
-0.0004605 0.00001
-0.001151 0.00001
-0.001842 0.00001
0 0.00001
-0.00046 0.00001
-0.00115 0.00001
-0.0002301 0.00001
-0.0004602 0.00001
```



Based on above tuning, mtry=3 is found to be the recommended, and mtree=1200 are the most optimal (with lowest OOB error). By using mtree=1200, I retry the simulation with mtry value 2, 3, 4, 5, 8, 10, 20, 50:

```
rf_tuneGrid <- expand.grid(.mtry = rf_mtry)
rf_metric <- "Accuracy"
rf_control <- trainControl(method="repeatedcv", number=10, repeats=3, search="random")

rf_fit <- randomForest(data=dat_train_simple, DEFAULT~.,
                       ntree=rf_mtree_optimal,
                       metric=rf_metric,
                       tuneGrid=rf_tuneGrid,
                       trControl=rf_control)

rf_predicted <- predict(rf_fit, dat_test)
rf_cm <- confusionMatrix(factor(rf_predicted),factor(dat_test$DEFAULT))
```

The above estimation reconfirmed that the best mtry=3. Based on these mtry=3 & mtree=1200, the result of estimation is as below:

Table 18: PREDICTION RESULT USING RANDOMFOREST

	x
Method	RandomForest
Accuracy	0.823029495084153
Balanced Accuracy	0.649740143969845

	x
Sensitivity	0.960624866252942
Specificity	0.338855421686747
F1	0.894223107569721

Following is the predicted Confusion Matrix:

```
                      Reference
Prediction             Actual Positive(1) Actual Negative(0)
  Predicted Positive(1)               4489                878
  Predicted Negative(0)                184                450
```

Following are list of important variables, according to Random Forest Method:

Table 19: MeanDecreaseGini

	MeanDecreaseGini
PAYDUE	657.967
PAYON_1	544.322
LIMIT_BAL	190.910
AGE_FAC	90.900
EDUCATION	46.579
PAYON_2	36.399
MARRIAGE	26.086
SEX	21.459
PAYON_3	9.912
PAYON_4	5.762
PAYON_5	4.945
PAYON_6	2.459

7.10 Gradient Boosting

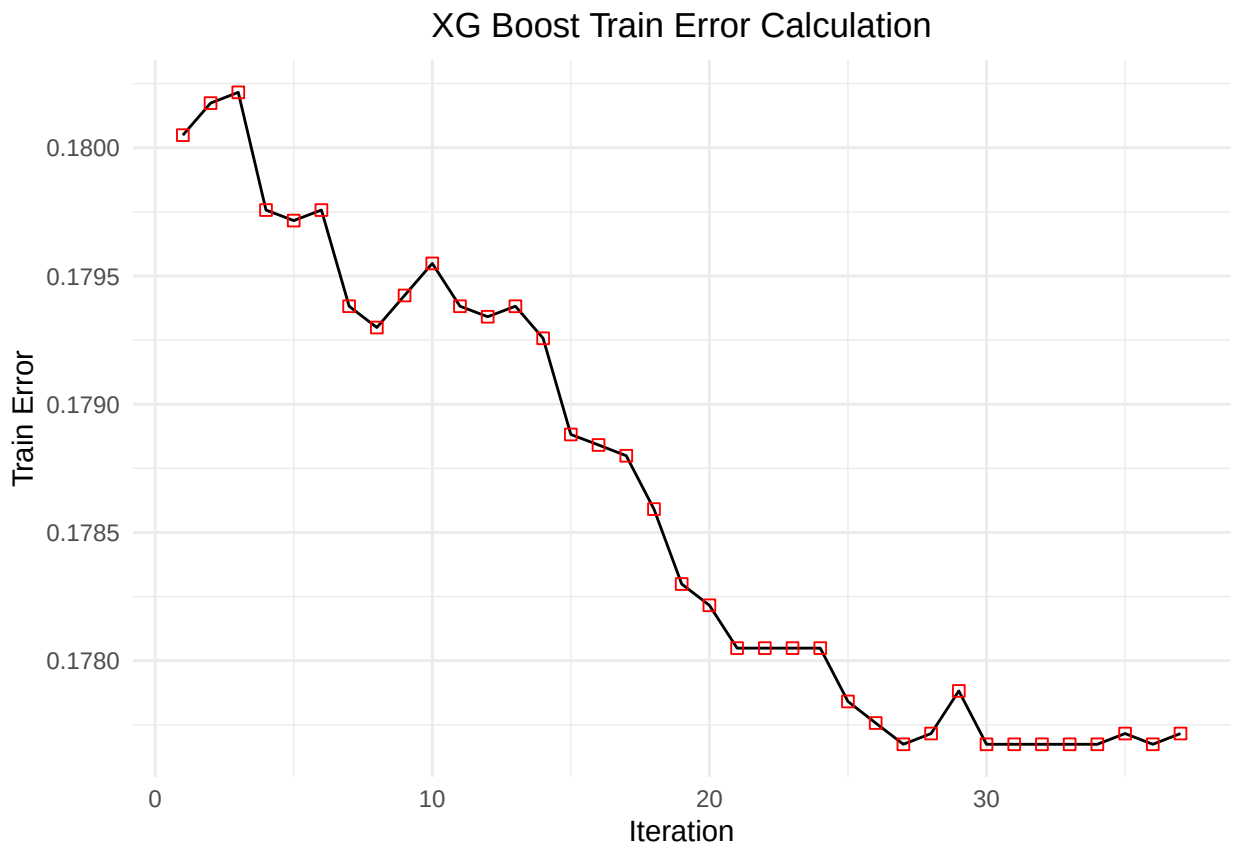
Modeling using Gradient Boosting method & the modeling result is as below:

```
dat_train_simple_x <- dat_train_simple %>% select(-DEFAULT) %>% data.matrix()
dat_train_simple_y <- dat_train_simple %>%
  mutate(DEFAULT=as.numeric(DEFAULT)-1) %>%
  select(DEFAULT) %>% data.matrix()


dat_test_simple_x <- dat_test_simple %>% select(-DEFAULT) %>% data.matrix()
dat_test_simple_y <- dat_test_simple %>%
  mutate(DEFAULT=as.numeric(DEFAULT)-1) %>%
  select(DEFAULT) %>% data.matrix()

# PERFORM XGBOOST MODELING
dat_train_m <- xgb.DMatrix(data = dat_train_simple_x, label= dat_train_simple_y)
dat_test_m <- xgb.DMatrix(data = dat_test_simple_x, label= dat_test_simple_y)

xg_fit <- xgboost(data = dat_train_m, # the data
                  max.depth = 5, # the maximum depth of each decision tree
                  nround = 1000, # number of boosting rounds
                  early_stopping_rounds = 10, # stop, if no improvement after rounds
                  gamma = 1, # add a regularization term
                  objective = "binary:logistic",
                  verbose=0)

xg_predicted <- predict(xg_fit, dat_test_m)
xg_cm <- confusionMatrix(factor(ifelse(xg_predicted>0.5,1,0)),factor(dat_test_simple_y))
```

Table 20: PREDICTION RESULT USING XGBOOST

	x
Method	XGBOOST
Accuracy	0.823529411764706
Balanced Accuracy	0.647635561505599
Sensitivity	0.963192809758185
Specificity	0.332078313253012
F1	0.89474207335255

XG Boost output above is obtained with the best XG Boost score 0.1777, with ntree = 27 iterations. Below is the train_error estimation of each iteration:



Following is the predicted Confusion Matrix:

```
                    Reference
Prediction           Actual Positive(1) Actual Negative(0)
  Predicted Positive(1)               4501                887
  Predicted Negative(0)                172                441
```

Following are list of important variables, according to XGBoost Method:

Table 21: XGBoost Variables Importancy

Feature	Gain	Cover	Frequency
PAYDUE	0.7509	0.1571	0.0844
LIMIT_BAL	0.1317	0.4515	0.3681
AGE_FAC	0.0497	0.1838	0.2638
EDUCATION	0.0262	0.0854	0.1135
SEX	0.0163	0.0374	0.0660
MARRIAGE	0.0134	0.0572	0.0644
PAYON_2	0.0057	0.0096	0.0184
PAYON_1	0.0041	0.0082	0.0107
PAYON_4	0.0015	0.0075	0.0061
PAYON_3	0.0005	0.0021	0.0046

8 Intepreting Result

8.1 Variables

There are total of 25 variables in the original dataset, and 36 variables in the final dataset I derived. I only uses following variables for my modeling,

```
 [1] "DEFAULT"   "LIMIT_BAL" "SEX"       "EDUCATION" "MARRIAGE"  "AGE_FAC"
 [7] "PAYDUE"    "PAYON_1"   "PAYON_2"   "PAYON_3"   "PAYON_4"   "PAYON_5"
[13] "PAYON_6"
```

and achieved the following output:

Table 22: PREDICTION RESULT

Method	Accuracy	Balanced	Sensitivity	Specificity	F1
GLM	0.8254	0.6491	0.9653	0.3328	0.8959
LM	0.8250	0.6502	0.9638	0.3366	0.8956
LOESS	0.8242	0.6500	0.9626	0.3373	0.8950
K-NN	0.7872	0.5356	0.9869	0.0843	0.8784
K-Means	0.2420	0.5052	0.0330	0.9774	0.0634
RF(3,1200)	0.8230	0.6497	0.9606	0.3389	0.8942
XGBOOST	0.8235	0.6476	0.9632	0.3321	0.8947

Comprehensive Dataset (dat_train_more)
In addition to the "Simplified dataset" *dat_train_simple* that contains 13 variables, I also run my modeling with the "Comprehensive Dataset" *dat_train_more* that prepared using following steps:

```
nzv <- nearZeroVar(dat_train)
dat_train_more <- dat_train %>%
  select(-all_of(nzv)) %>%
  select(-c(EDU_text, SEX_text, MAR_text, DEF_text)) %>%
  select(-AGE)
```

The full dataset is first examined by removing "Near Zero Variable", following by removing "Convenience Variables" and lastly the AGE variable that superseded by AGE_FAC. The Comprehensive Dataset consists of following 26 variables:

```
 [1] "LIMIT_BAL" "SEX"       "EDUCATION" "MARRIAGE"  "PAY_1"     "PAY_2"
 [7] "PAY_3"     "PAY_4"     "PAY_5"     "PAY_6"     "BILLED_1"  "BILLED_2"
[13] "BILLED_3"  "BILLED_4"  "BILLED_5"  "BILLED_6"  "PAYMENT_1" "PAYMENT_2"
[19] "PAYMENT_3" "PAYMENT_4" "PAYMENT_5" "PAYMENT_6" "DEFAULT"   "AGE_FAC"
[25] "PAYON_1"   "PAYDUE"
```

The modeling output with Comprehensive Dataset ias as below:

Modeling using Comprehensive Dataset is performed outside of generation of this document. The full script for performing the modeling is available at APPENDIX A.

Table 23: PREDICTION RESULT (COMPREHENSIVE DATASET)

Method	Accuracy	Balanced	Sensitivity	Specificity	F1
GLM	0.8230	0.6505	0.9600	0.3411	0.8942
LM	0.8240	0.6550	0.9583	0.3517	0.8945
LOESS	0.8225	0.6481	0.9611	0.3351	0.8940
K-NN	0.7757	0.5326	0.9688	0.0964	0.8706
K-Means	0.2430	0.5010	0.0381	0.9639	0.0727
RF(3,800)	0.8210	0.6568	0.9514	0.3622	0.8922
XGBOOST	0.8210	0.6609	0.9482	0.3735	0.8919

Comparing the achieved modeling output of "Simplified dataset", the output from "Comprehensive Dataset" is less preferred, as shown in following comparison table. As such, I used the simplified dataset result as my primary output.

Table 24: COMPARE DATASET PREDICTION RESULT

	GLM	LM	LOESS	K-NN	K-Means	RF(3,1200)	XGBOOST
Accuracy	0.8230	0.8240	0.8225	0.7757	0.2430	0.8210	0.8210
Balanced	0.6505	0.6550	0.6481	0.5326	0.5010	0.6568	0.6609
Sensitivity	0.9600	0.9583	0.9611	0.9688	0.0381	0.9514	0.9482
Specificity	0.3411	0.3517	0.3351	0.0964	0.9639	0.3622	0.3735
F1	0.8942	0.8945	0.8940	0.8706	0.0727	0.8922	0.8919
DiffAccuracy	-0.0023	-0.0010	-0.0017	-0.0115	0.0010	-0.0020	-0.0025
DiffBalanced	0.0015	0.0047	-0.0019	-0.0031	-0.0042	0.0071	0.0132
DiffSensitivity	-0.0053	-0.0056	-0.0015	-0.0182	0.0051	-0.0092	-0.0150
DiffSpecificity	0.0083	0.0151	-0.0023	0.0120	-0.0136	0.0233	0.0414
DiffF1	-0.0018	-0.0011	-0.0010	-0.0078	0.0093	-0.0020	-0.0028

8.2 Metric Selection

There are few metrics as described in previous section that my models have computed and reported. Some models scored high in some metrics, and some others scored high in another. Selection of metric is important to present a more accurate perspective of my analysis outcome.

Mathematically, Accuracy is the sum of correct prediction outcome over total populations, sensitivity and specificity focusing in either True Positive or True Negative, and F-Score is the harmonic mean of sensitivity and specificity.

Accuracy is always the preferred metric for categorical analysis, however, as the UCI Credit Card Default dataset is imbalance with only 22.12% "Positive" output (DEFAULT=1), accuracy is not suitable as it does not take into consideration of prevalence of "Positive" output.

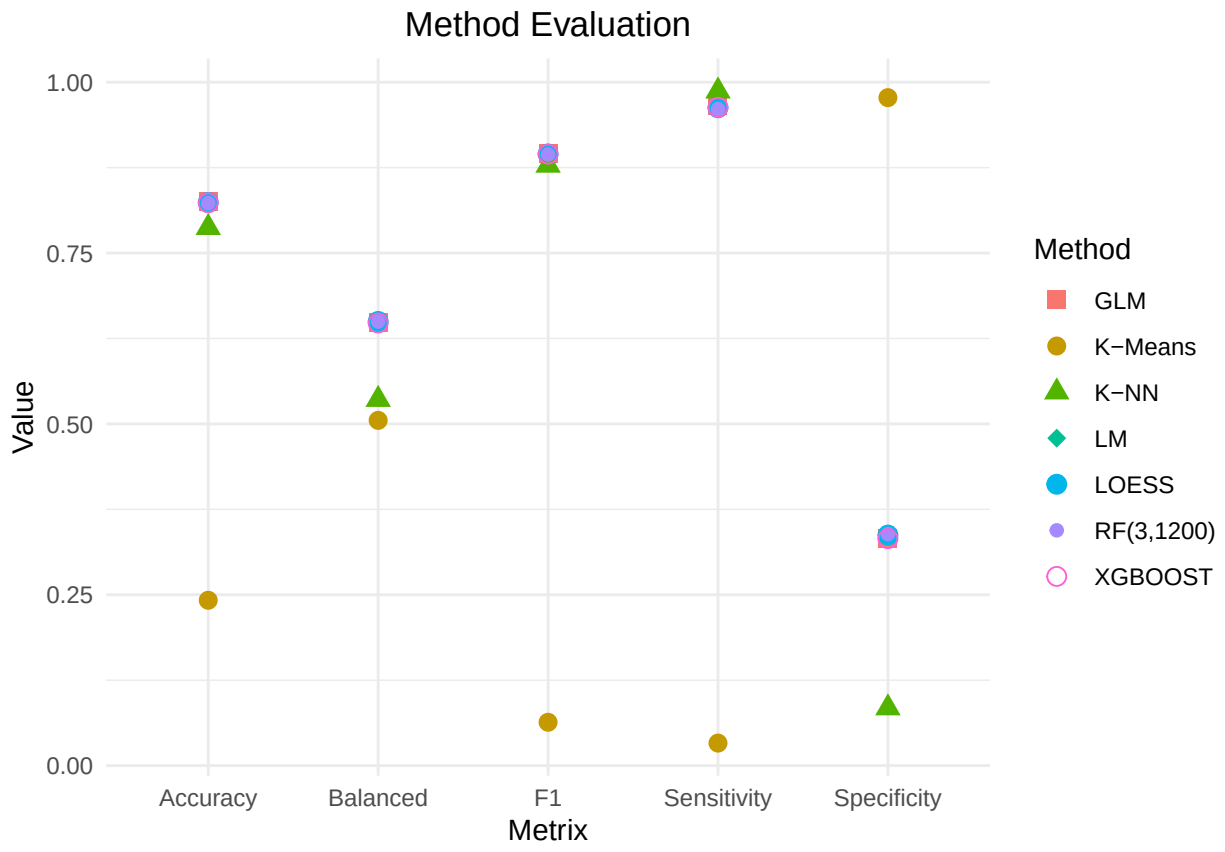
My focus is in predicting the overall possibility of a client Defaulting on OCT 2005 credit card payment, not leaning toward True Positive and True Negative, thus, the balance of these two should be the best metric. As such, I chosen **F-Score** as my primary metric.

8.3 Result Analysis

As shown in below table and chart, F-Score value for GLM, LM, LOESS, RandomForest & XGBoost are quite close.

Table 25: PREDICTION RESULT

Method	Accuracy	Balanced	Sensitivity	Specificity	F1
GLM	0.8254	0.6491	0.9653	0.3328	0.8959
LM	0.8250	0.6502	0.9638	0.3366	0.8956
LOESS	0.8242	0.6500	0.9626	0.3373	0.8950
K-NN	0.7872	0.5356	0.9869	0.0843	0.8784
K-Means	0.2420	0.5052	0.0330	0.9774	0.0634
RF(3,1200)	0.8230	0.6497	0.9606	0.3389	0.8942
XGBOOST	0.8235	0.6476	0.9632	0.3321	0.8947



From the observation above, I conclude that **GLM method** is the best method for predicting a client probability of defaulting Oct 2005 Credit Card billing payment, by having **highest F-Score value of 0.8959**.

8.4 Other Methods

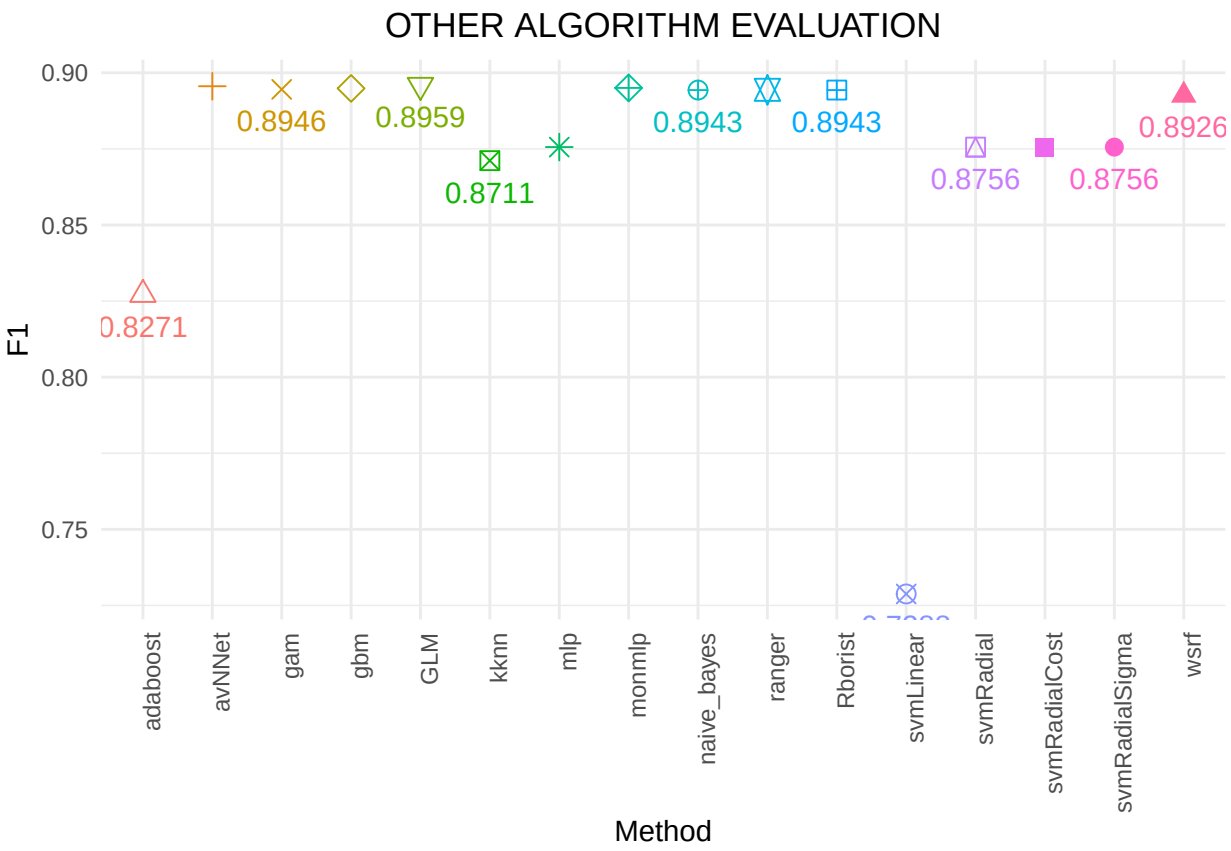
In addition to above mentioned methods, I also perform estimation on following methods:

- Naive_bayes
- SVMLinear
- kknn
- GAM
- Ranger
- WSRF
- Rborist
- avNNet
- MLP
- monMLP
- GBM
- ADABoost
- SVMRadial
- SVMRadialCost
- SVMRadialSigma

Comparing the achieved F-Score on GLM method with other models, GLM method still output the highest F-Score.

Table 26: PREDICTION RESULT (OTHER ALGORITHMS)

Method	Accuracy	Balanced	Sensitivity	Specificity	F1
GLM	0.8254	0.6491	0.9653	0.3328	0.8959
naive_bayes	0.8235	0.6530	0.9589	0.3471	0.8943
svmLinear	0.6231	0.5887	0.6503	0.5271	0.7288
kknn	0.7877	0.6192	0.9215	0.3170	0.8711
gam	0.8235	0.6501	0.9613	0.3389	0.8946
ranger	0.8234	0.6505	0.9606	0.3404	0.8944
wsrf	0.8189	0.6328	0.9666	0.2989	0.8926
Rborist	0.8235	0.6530	0.9589	0.3471	0.8943
avNNet	0.8254	0.6537	0.9617	0.3456	0.8956
mlp	0.7787	0.5000	1.0000	0.0000	0.8756
monmlp	0.8245	0.6531	0.9606	0.3456	0.8950
gbm	0.8242	0.6521	0.9608	0.3434	0.8949
adaboost	0.7342	0.6304	0.8166	0.4443	0.8271
svmRadial	0.7787	0.5000	1.0000	0.0000	0.8756
svmRadialCost	0.7787	0.5000	1.0000	0.0000	0.8756
svmRadialSigma	0.7787	0.5000	1.0000	0.0000	0.8756


OTHER ALGORITHM EVALUATION
F1
0.90
0.85
0.80
0.75
0.8271
0.8946
0.8959
0.8711
0.8943
0.8943
0.8756
0.8756
0.8926
adaboost
avNNet
gam
gbm
GLM
kknn
mlp
monmlp
naive_bayes
ranger
Rborist
svmLinear
svmRadial
svmRadialCost
svmRadialSigma
wsrf
Method

9 Conclusion

The objective of this report is to apply the knowledge acquired throughout the Harvardx Data Science courses, to analyze the selected UCR Credit Card Default dataset (downloaded from kaggle.com) using Recommendation System approach, and to suggest the best model to predict a client probability of defaulting Oct 2005 credit card bill paymet. To ensure the probability of accurate analysis, I selected and followed the OSEMN Framework standardized process. The process consists of 5 steps, (1) Obtain Data, (2) Scrub Data, (3) Explore Data, (4) Model Data and (5) Intepreting Data.

After evaluating several models and with 2 difference variables set, the **Generalized Linear Model** is found to be the best with **F-score of 89.59%** for predicting the probability of a client defauling on Oct 2005 Credit Card payment, with the "Simplified Dataset" with following variables:

```
 [1] "DEFAULT"   "LIMIT_BAL" "SEX"       "EDUCATION" "MARRIAGE"  "AGE_FAC"
 [7] "PAYDUE"    "PAYON_1"   "PAYON_2"   "PAYON_3"   "PAYON_4"   "PAYON_5"
[13] "PAYON_6"
```

9.1 Limitation

There are few limitation encounters in research, analyze, training and choosing models for the project modeling:

Hardware limitation: The training and modeling are perform on a laptop that has limited memory and processing powers. I have attempted to evaluate others models, but all of them failed due to hardware limitation. This limitation can be overcome with a better hardware or machine.

Algorithm limitation: According to kaggle.com website, the are few modeling that may present a better result. Most of these modeling are performed on specialize program. Apart of these algorithm, there's also other R package that may resulted better accuracy. This limitation can be overcome with more details exploration and research.

Quesionable Dataset: As discuss in Data Explorarion section, the PAYMENT_x, BILLED_x and BAL_LIMIT variables are contradict and do not make sense from the perspective from Credit Card industry practice. As such, I made decision to use only BAL_LIMIT and drop the other set of variables. Inclusive of all variables include above has also resulted a less preferred F-Score value, compare with the dataset with less variables as discussed. The evaluation can improved if a better dataset or a clear definition is provided on above notice abnormality.

9.2 Future Works

More research, analysis, and training of models are needed to get the best result in predicting client defaulting. Following modeling techniques warrants further studies to improve the overall accuracy,

Appendix A: Modeling With "Comprehensive Dataset"

Following is the full script for executing modeling using "Comprehensive dataset" dat_train_more:

```
#.......................... . ..........................#
#    COMPREHENSIVE DATASET
#    - Preparing Script
#.......................... . ..........................#

sink(paste0("modeling-dat_train_more-",format(Sys.Date(),"%Y%m%d"),".log"),
     append=FALSE, split=TRUE)
Sys.Date()

nzv <- nearZeroVar(dat_train)
dat_train_more <- dat_train %>%
  select(-all_of(nzv)) %>%
  select(-c(EDU_text, SEX_text, MAR_text, DEF_text)) %>%
  select(-AGE)

dat_test_more <- dat_test %>%
  select(-all_of(nzv)) %>%
  select(-c(EDU_text, SEX_text, MAR_text, DEF_text)) %>%
  select(-AGE)

#--------------------------- . ---------------------------#

result_more <- tibble(Method = "",
                      Accuracy = 0,
                      Balanced = 0,
                      Sensitivity = 0,
                      Specificity = 0,
                      F1 = 0)

#--------------------------- . ---------------------------#
#    PREDICT USING LOGISTIC REGRESSION
#--------------------------- . ---------------------------#

glm_fit <- train(data=dat_train_more, DEFAULT~.,method = "glm")
glm_predicted <- predict(glm_fit, newdata = dat_test)
glm_cm <- confusionMatrix(factor(glm_predicted),factor(dat_test$DEFAULT))

result_more <- rbind(result_more, with(glm_cm,
                                       c(Method = "GLM",
                                         Accuracy = overall["Accuracy"],
                                         Balanced = byClass["Balanced Accuracy"],
                                         Sensitivity = byClass["Sensitivity"],
                                         Specificity = byClass["Specificity"],
                                         F1 = byClass["F1"])))
result_more %>% knitr::kable(caption="PREDICTION ACCURACY", digits=4)

#--------------------------- . ---------------------------#
#    PREDICT USING LM
#--------------------------- . ---------------------------#
lm_fit <- lm(DEFAULT~.,
```

```
             data=dat_train_more)
lm_predict <- predict(lm_fit, dat_test)
lm_predicted <- ifelse(lm_predict>1.5,1,0 )
lm_cm <- confusionMatrix(factor(lm_predicted),factor(dat_test$DEFAULT))

result_more <- rbind(result_more, with(lm_cm,
                                       c(Method = "LM",
                                         Accuracy = overall["Accuracy"],
                                         Balanced = byClass["Balanced Accuracy"],
                                         Sensitivity = byClass["Sensitivity"],
                                         Specificity = byClass["Specificity"],
                                         F1 = byClass["F1"])))
result_more %>% knitr::kable(caption="PREDICTION ACCURACY", digits=4)

#---------------------------- . ----------------------------#
#    PREDICT LOESS
#---------------------------- . ----------------------------#
loess_fit <- train(data=dat_train_more, DEFAULT~., method = "gamLoess")
loess_predicted <- predict(loess_fit, dat_test)
loess_cm <- confusionMatrix(factor(loess_predicted),factor(dat_test$DEFAULT))

result_more <- rbind(result_more, with(loess_cm,
                                       c(Method = "LOESS",
                                         Accuracy = overall["Accuracy"],
                                         Balanced = byClass["Balanced Accuracy"],
                                         Sensitivity = byClass["Sensitivity"],
                                         Specificity = byClass["Specificity"],
                                         F1 = byClass["F1"])))
result_more %>% knitr::kable(caption="PREDICTION ACCURACY", digits=4)



#---------------------------- . ----------------------------#
#    PREDICT KNN
#---------------------------- . ----------------------------#
set.seed(1,sample.kind = "Rounding")
knn_control <- trainControl(method = "repeatedcv", number = 10, p = .9, repeats=10)
knn_fit <- train(data=dat_train_more, DEFAULT~., method = "knn",
                 tuneGrid = data.frame(k = seq(3, 21, by=2)),
                 trControl=knn_control)
knn_predicted <- predict(knn_fit, dat_test)
knn_cm <- confusionMatrix(factor(knn_predicted),factor(dat_test$DEFAULT))

result_more <- rbind(result_more, with(knn_cm,
                                       c(Method = "K-NN",
                                         Accuracy = overall["Accuracy"],
                                         Balanced = byClass["Balanced Accuracy"],
                                         Sensitivity = byClass["Sensitivity"],
                                         Specificity = byClass["Specificity"],
                                         F1 = byClass["F1"])))
result_more %>% knitr::kable(caption="PREDICTION ACCURACY", digits=4)
```

```r
#---------------------------- . -----------------------------#
#    PREDICT K-MEANS
#---------------------------- . -----------------------------#
set.seed(1, sample.kind = "Rounding")
f_predictkmeans <- function(x, k) {
  centers <- k$centers    # extract cluster centers
  # calculate distance to cluster centers
  distances <- sapply(1:nrow(x), function(i){
    apply(centers, 1, function(y) dist(rbind(x[i,], y)))
  })
  max.col(-t(distances))  # select cluster with min distance to center
}
k_means <- kmeans(dat_train_more[,-1],centers=2)
k_predict <- f_predictkmeans(dat_test,k_means)
k_predicted <- ifelse(k_predict>1.5,0,1)
k_cm <- confusionMatrix(factor(k_predicted),factor(dat_test$DEFAULT))

result_more <- rbind(result_more, with(k_cm,
                                       c(Method = "K-Means",
                                         Accuracy = overall["Accuracy"],
                                         Balanced = byClass["Balanced Accuracy"],
                                         Sensitivity = byClass["Sensitivity"],
                                         Specificity = byClass["Specificity"],
                                         F1 = byClass["F1"])))
result_more %>% knitr::kable(caption="PREDICTION ACCURACY", digits=4)


#---------------------------- . -----------------------------#
#    PREDICT USING RANDOM FOREST
#    - mtry: Number of variables randomly sampled as candidates at each split.
#    - ntree: Number of trees to grow.
#    - various mtry & ntree values combination are explored to reach the range as below
#---------------------------- . -----------------------------#

# First, find the best parameter for RF
#
# set.seed(1,sample.kind = "Rounding")
#
# rf_ntrees <- seq(200,by=200, len=10)
# rf_tuning <- sapply(rf_ntrees, function(nt){
#   t_mtry <- tuneRF(dat_train_more[,-1],
#                    dat_train_more$DEFAULT,
#                    stepFactor=1.5, improve=1e-5, ntree=nt,
#                    plot=TRUE,trace=FALSE)
#   t_mtry
# })
#
#
# rf_tuningT <- cbind(rf_tuning[[x1]][1:3,],ntree=rep(3,rf_ntrees[x]))
# rep(2:10,function(x){
#     rf_tuningT <- rbind(rf_tuningT, cbind(rf_tuning[[x]][1:3,],ntree=rep(3,rf_ntrees[x])))
# })
#
```

```r
# x=5
# rf_tuningT <- rbind(rf_tuningT, cbind(rf_tuning[[x]][1:3,],ntree=rep(3,rf_ntrees[x])))
# rf_ntrees[4]
# rf_tuning
# rownames(rf_tuning) <- c(2,3,4)
# colnames(rf_tuning) <- rf_ntrees
# rf_tuning <- cbind(rf_tuning,mtry=c(2,3,4))
#
# # t_mtry <- tuneRF(dat_train_more[,-1],
# #                  dat_train_more$DEFAULT,
# #                  stepFactor=1.5, improve=1e-5, ntreeTry=1200,
# #                  plot=TRUE,trace=TRUE,doBest=TRUE)
#
# rf_tuning %>% as.data.frame() %>% gather("ntree","value",-mtry) %>%
#   mutate(ntree=str_sub(paste0("000",ntree),-6,-1)) %>%
#   mutate(mtry=as.factor(mtry)) %>%
#   ggplot(aes(mtry,value)) + geom_point() +
#   ggtitle("RANDOM FOREST MTRY & NTREE PARAMETERS TUNNING") +
#   ylab("OOB ERROR") +
#   geom_hline(yintercept = 0.18125,col="blue", linetype="dashed", size=0.5) +
#   geom_vline(xintercept = 2,col="blue", linetype="dashed", size=0.5) +
#   theme_minimal() + theme(plot.title = element_text(hjust = 0.5)) +
#   facet_wrap(~ntree, strip.position="bottom")

#----------------
# Based on above chart, as mtry=3 is more consistent as recommended,
# I selected mtry=3, ntree=800, mtry=3 is further confirm with following

rf_tuneGrid <- expand.grid(.mtry = c(2,3,4,5,8,10,20,50))
rf_metric <- "Accuracy"
rf_control <- trainControl(method="repeatedcv", number=10, repeats=3, search="random")

rf_fit <- randomForest(data=dat_train_more, DEFAULT~.,
                       ntree=800,
                       metric=rf_metric,
                       tuneGrid=rf_tuneGrid,
                       trControl=rf_control)

rf_predicted <- predict(rf_fit, dat_test)
rf_cm <- confusionMatrix(factor(rf_predicted),factor(dat_test$DEFAULT))

result_more <- rbind(result_more, with(rf_cm,
                                       c(Method = "RF(3,800)",
                                         Accuracy = overall["Accuracy"],
                                         Balanced = byClass["Balanced Accuracy"],
                                         Sensitivity = byClass["Sensitivity"],
                                         Specificity = byClass["Specificity"],
                                         F1 = byClass["F1"])))
result_more %>% knitr::kable(caption="PREDICTION ACCURACY", digits=4)


# ------------------------------------------------------
# STUDY IMPORTANT of VARIABLE
```

```r
importance(rf_fit)%>% as.data.frame() %>%
  arrange(-MeanDecreaseGini) %>%
  knitr::kable(caption="MeanDecreaseGini")




#---------------------------- . -----------------------------#
#    PREDICT USING XGBOOST
#---------------------------- . -----------------------------#

dat_train_more_x <- dat_train_more %>% select(-DEFAULT) %>% data.matrix()
dat_train_more_y <- dat_train_more %>%
  mutate(DEFAULT=as.numeric(DEFAULT)-1) %>%
  select(DEFAULT) %>% data.matrix()


dat_test_more_x <- dat_test_more %>% select(-DEFAULT) %>% data.matrix()
dat_test_more_y <- dat_test_more %>%
  mutate(DEFAULT=as.numeric(DEFAULT)-1) %>%
  select(DEFAULT) %>% data.matrix()

# PERFORM XGBOOST MODELING
dat_train_m <- xgb.DMatrix(data = dat_train_more_x, label= dat_train_more_y)
dat_test_m <- xgb.DMatrix(data = dat_test_more_x, label= dat_test_more_y)

xg_fit <- xgboost(data = dat_train_m, # the data
                  max.depth = 5, # the maximum depth of each decision tree
                  nround = 1000, # number of boosting rounds
                  early_stopping_rounds = 10, # if we dont see an improvement in this many rounds, stop
                  #scale_pos_weight = sum(dat_train_simple_y==1)/sum(dat_train_simple_y==0), # control
                  gamma = 1, # add a regularization term
                  objective = "binary:logistic")

xg_predicted <- predict(xg_fit, dat_test_m)
xg_cm <- confusionMatrix(factor(ifelse(xg_predicted>0.5,1,0)),factor(dat_test_more_y))

xgb.importance(model=xg_fit)


result_more <- rbind(result_more, with(xg_cm,
                                       c(Method = "XGBOOST",
                                         Accuracy = overall["Accuracy"],
                                         Balanced = byClass["Balanced Accuracy"],
                                         Sensitivity = byClass["Sensitivity"],
                                         Specificity = byClass["Specificity"],
                                         F1 = byClass["F1"])))
result_more %>% knitr::kable(caption="PREDICTION RESULT", digits=4)

#---------------------------- . -----------------------------#
#    CONSOLIDATE PREDICTED RESULT
#---------------------------- . -----------------------------#
```

```r
result_more %>% knitr::kable(caption="PREDICTION RESULT", digits=4, format.args=list(big.mark=","))
```

Appendix B: "Modeling Using Other Methods"

Following are the full script for modeling using following methods:

```
#.......................... . ..........................#
#    SIMPLIFIED DATASET
#.......................... . ..........................#

sink(paste0("modeling-dat_train-other-",format(Sys.Date(),"%Y%m%d"),".log"),
     append=FALSE, split=TRUE)

# [1] "DEFAULT"   "LIMIT_BAL" "SEX"       "EDUCATION" "MARRIAGE"  "AGE_FAC"   "PAYDUE"    "PAYON_1"
# [9] "PAYON_2"   "PAYON_3"   "PAYON_4"   "PAYON_5"   "PAYON_6"

dat_train_simple <- dat_train %>%
  select(DEFAULT, LIMIT_BAL, SEX, EDUCATION, MARRIAGE, AGE_FAC, PAYDUE,
         PAYON_1, PAYON_2, PAYON_3, PAYON_4, PAYON_5, PAYON_6)

dat_test_simple <- dat_test %>%
  select(DEFAULT, LIMIT_BAL, SEX, EDUCATION, MARRIAGE, AGE_FAC, PAYDUE,
         PAYON_1, PAYON_2, PAYON_3, PAYON_4, PAYON_5, PAYON_6)

names(dat_train_simple)    #13 variables

#--------------------------- . ---------------------------#

result_other <- tibble(Method = "",
                       Accuracy = 0,
                       Balanced = 0,
                       Sensitivity = 0,
                       Specificity = 0,
                       F1 = 0)


names(dat_train_simple)

############################## . ##############################
#    MODELING USING OTHER APPROACH
#    - USE DEFAULT PARAMETER
#    - "naive_bayes", "svmLinear",  "kknn", "loclda", "gam",
#      "ranger","wsrf", "Rborist", "avNNet", "mlp", "monmlp", "gbm",
#      "adaboost", "svmRadial", "svmRadialCost", "svmRadialSigma"
############################## . ##############################


#--------------------------- . ---------------------------#
#    PREDICT USING naive_bayes
#--------------------------- . ---------------------------#
nb_fit <- train(data=dat_train_simple, DEFAULT~.,
                method = "naive_bayes")
nb_predicted <- predict(nb_fit, dat_test)
nb_cm <- confusionMatrix(factor(nb_predicted),factor(dat_test$DEFAULT))

result_other <- rbind(result_other, with(nb_cm,
```

```
                                          c(Method = "naive_bayes",
                                            Accuracy = overall["Accuracy"],
                                            Balanced = byClass["Balanced Accuracy"],
                                            Sensitivity = byClass["Sensitivity"],
                                            Specificity = byClass["Specificity"],
                                            F1 = byClass["F1"])))
result_other %>% knitr::kable(caption="PREDICTION RESULT", digits=4)


#---------------------------- . -----------------------------#
#    PREDICT USING svmLinear
#---------------------------- . -----------------------------#
svml_fit <- train(data=dat_train_simple, DEFAULT~.,
                  method = "svmLinear")
svml_predicted <- predict(svml_fit, dat_test)
svml_cm <- confusionMatrix(factor(svml_predicted),factor(dat_test$DEFAULT))

result_other <- rbind(result_other, with(svml_cm,
                                          c(Method = "svmLinear",
                                            Accuracy = overall["Accuracy"],
                                            Balanced = byClass["Balanced Accuracy"],
                                            Sensitivity = byClass["Sensitivity"],
                                            Specificity = byClass["Specificity"],
                                            F1 = byClass["F1"])))
result_other %>% knitr::kable(caption="PREDICTION RESULT", digits=4)

#---------------------------- . -----------------------------#
#    PREDICT USING kknn
#---------------------------- . -----------------------------#
kknn_fit <- train(data=dat_train_simple, DEFAULT~.,
                  method = "kknn")
kknn_predicted <- predict(kknn_fit, dat_test)
kknn_cm <- confusionMatrix(factor(kknn_predicted),factor(dat_test$DEFAULT))

result_other <- rbind(result_other, with(kknn_cm,
                                          c(Method = "kknn",
                                            Accuracy = overall["Accuracy"],
                                            Balanced = byClass["Balanced Accuracy"],
                                            Sensitivity = byClass["Sensitivity"],
                                            Specificity = byClass["Specificity"],
                                            F1 = byClass["F1"])))
result_other %>% knitr::kable(caption="PREDICTION RESULT", digits=4)


#---------------------------- . -----------------------------#
#    PREDICT USING gam
#---------------------------- . -----------------------------#
gam_fit <- train(data=dat_train_simple, DEFAULT~.,
                 method = "gam")
gam_predicted <- predict(gam_fit, dat_test)
gam_cm <- confusionMatrix(factor(gam_predicted),factor(dat_test$DEFAULT))

result_other <- rbind(result_other, with(gam_cm,
```

```
                                          c(Method = "gam",
                                            Accuracy = overall["Accuracy"],
                                            Balanced = byClass["Balanced Accuracy"],
                                            Sensitivity = byClass["Sensitivity"],
                                            Specificity = byClass["Specificity"],
                                            F1 = byClass["F1"])))
result_other %>% knitr::kable(caption="PREDICTION RESULT", digits=4)


#---------------------------- . -----------------------------#
#    PREDICT USING ranger
#---------------------------- . -----------------------------#
ranger_fit <- train(data=dat_train_simple, DEFAULT~.,
                    method = "ranger")
ranger_predicted <- predict(ranger_fit, dat_test)
ranger_cm <- confusionMatrix(factor(ranger_predicted),factor(dat_test$DEFAULT))

result_other <- rbind(result_other, with(ranger_cm,
                                          c(Method = "ranger",
                                            Accuracy = overall["Accuracy"],
                                            Balanced = byClass["Balanced Accuracy"],
                                            Sensitivity = byClass["Sensitivity"],
                                            Specificity = byClass["Specificity"],
                                            F1 = byClass["F1"])))
result_other %>% knitr::kable(caption="PREDICTION RESULT", digits=4)


#---------------------------- . -----------------------------#
#    PREDICT USING wsrf
#---------------------------- . -----------------------------#
wsrf_fit <- train(data=dat_train_simple, DEFAULT~.,
                  method = "wsrf")
wsrf_predicted <- predict(wsrf_fit, dat_test)
wsrf_cm <- confusionMatrix(factor(wsrf_predicted),factor(dat_test$DEFAULT))

result_other <- rbind(result_other, with(wsrf_cm,
                                          c(Method = "wsrf",
                                            Accuracy = overall["Accuracy"],
                                            Balanced = byClass["Balanced Accuracy"],
                                            Sensitivity = byClass["Sensitivity"],
                                            Specificity = byClass["Specificity"],
                                            F1 = byClass["F1"])))
result_other %>% knitr::kable(caption="PREDICTION RESULT", digits=4)


#---------------------------- . -----------------------------#
#    PREDICT USING Rborist
#---------------------------- . -----------------------------#
Rbor_fit <- train(data=dat_train_simple, DEFAULT~.,
                  method = "Rborist")
Rbor_predicted <- predict(Rbor_fit, dat_test)
Rbor_cm <- confusionMatrix(factor(Rbor_predicted),factor(dat_test$DEFAULT))
```

```
result_other <- rbind(result_other, with(Rbor_cm,
                                         c(Method = "Rborist",
                                           Accuracy = overall["Accuracy"],
                                           Balanced = byClass["Balanced Accuracy"],
                                           Sensitivity = byClass["Sensitivity"],
                                           Specificity = byClass["Specificity"],
                                           F1 = byClass["F1"])))
result_other %>% knitr::kable(caption="PREDICTION RESULT", digits=4)



#---------------------------- . -----------------------------#
#    PREDICT USING avNNet
#---------------------------- . -----------------------------#
avNNet_fit <- train(data=dat_train_simple, DEFAULT~.,
                    method = "avNNet")
avNNet_predicted <- predict(avNNet_fit, dat_test)
avNNet_cm <- confusionMatrix(factor(avNNet_predicted),factor(dat_test$DEFAULT))

result_other <- rbind(result_other, with(avNNet_cm,
                                         c(Method = "avNNet",
                                           Accuracy = overall["Accuracy"],
                                           Balanced = byClass["Balanced Accuracy"],
                                           Sensitivity = byClass["Sensitivity"],
                                           Specificity = byClass["Specificity"],
                                           F1 = byClass["F1"])))
result_other %>% knitr::kable(caption="PREDICTION RESULT", digits=4)


#---------------------------- . -----------------------------#
#    PREDICT USING mlp
#---------------------------- . -----------------------------#
mlp_fit <- train(data=dat_train_simple, DEFAULT~.,
                 method = "mlp")
mlp_predicted <- predict(mlp_fit, dat_test)
mlp_cm <- confusionMatrix(factor(mlp_predicted),factor(dat_test$DEFAULT))

result_other <- rbind(result_other, with(mlp_cm,
                                         c(Method = "mlp",
                                           Accuracy = overall["Accuracy"],
                                           Balanced = byClass["Balanced Accuracy"],
                                           Sensitivity = byClass["Sensitivity"],
                                           Specificity = byClass["Specificity"],
                                           F1 = byClass["F1"])))
result_other %>% knitr::kable(caption="PREDICTION RESULT", digits=4)


#---------------------------- . -----------------------------#
#    PREDICT USING monmlp
#---------------------------- . -----------------------------#
monmlp_fit <- train(data=dat_train_simple, DEFAULT~.,
                    method = "monmlp")
monmlp_predicted <- predict(monmlp_fit, dat_test)
```

```
monmlp_cm <- confusionMatrix(factor(monmlp_predicted),factor(dat_test$DEFAULT))

result_other <- rbind(result_other, with(monmlp_cm,
                                         c(Method = "monmlp",
                                           Accuracy = overall["Accuracy"],
                                           Balanced = byClass["Balanced Accuracy"],
                                           Sensitivity = byClass["Sensitivity"],
                                           Specificity = byClass["Specificity"],
                                           F1 = byClass["F1"])))
result_other %>% knitr::kable(caption="PREDICTION RESULT", digits=4)


#---------------------------- . -----------------------------#
#    PREDICT USING gbm
#---------------------------- . -----------------------------#
gbm_fit <- train(data=dat_train_simple, DEFAULT~.,
                 method = "gbm")
gbm_predicted <- predict(gbm_fit, dat_test)
gbm_cm <- confusionMatrix(factor(gbm_predicted),factor(dat_test$DEFAULT))

result_other <- rbind(result_other, with(gbm_cm,
                                         c(Method = "gbm",
                                           Accuracy = overall["Accuracy"],
                                           Balanced = byClass["Balanced Accuracy"],
                                           Sensitivity = byClass["Sensitivity"],
                                           Specificity = byClass["Specificity"],
                                           F1 = byClass["F1"])))
result_other %>% knitr::kable(caption="PREDICTION RESULT", digits=4)


#---------------------------- . -----------------------------#
#    PREDICT USING adaboost
#---------------------------- . -----------------------------#
adab_fit <- train(data=dat_train_simple, DEFAULT~.,
                  method = "adaboost")
adab_predicted <- predict(adab_fit, dat_test)
adab_cm <- confusionMatrix(factor(adab_predicted),factor(dat_test$DEFAULT))

result_other <- rbind(result_other, with(adab_cm,
                                         c(Method = "adaboost",
                                           Accuracy = overall["Accuracy"],
                                           Balanced = byClass["Balanced Accuracy"],
                                           Sensitivity = byClass["Sensitivity"],
                                           Specificity = byClass["Specificity"],
                                           F1 = byClass["F1"])))
result_other %>% knitr::kable(caption="PREDICTION RESULT", digits=4)


#---------------------------- . -----------------------------#
#    PREDICT USING svmRadial
#---------------------------- . -----------------------------#
svmr_fit <- train(data=dat_train_simple, DEFAULT~.,
                  method = "svmRadial")
```

```
svmr_predicted <- predict(svmr_fit, dat_test)
svmr_cm <- confusionMatrix(factor(svmr_predicted),factor(dat_test$DEFAULT))

result_other <- rbind(result_other, with(svmr_cm,
                                         c(Method = "svmRadial",
                                           Accuracy = overall["Accuracy"],
                                           Balanced = byClass["Balanced Accuracy"],
                                           Sensitivity = byClass["Sensitivity"],
                                           Specificity = byClass["Specificity"],
                                           F1 = byClass["F1"])))
result_other %>% knitr::kable(caption="PREDICTION RESULT", digits=4)


#---------------------------- . ----------------------------#
#    PREDICT USING svmRadialCost
#---------------------------- . ----------------------------#
svmrc_fit <- train(data=dat_train_simple, DEFAULT~.,
                   method = "svmRadialCost")
svmrc_predicted <- predict(svmrc_fit, dat_test)
svmrc_cm <- confusionMatrix(factor(svmrc_predicted),factor(dat_test$DEFAULT))

result_other <- rbind(result_other, with(svmrc_cm,
                                         c(Method = "svmRadialCost",
                                           Accuracy = overall["Accuracy"],
                                           Balanced = byClass["Balanced Accuracy"],
                                           Sensitivity = byClass["Sensitivity"],
                                           Specificity = byClass["Specificity"],
                                           F1 = byClass["F1"])))
result_other %>% knitr::kable(caption="PREDICTION RESULT", digits=4)


#---------------------------- . ----------------------------#
#    PREDICT USING svmRadialSigma
#---------------------------- . ----------------------------#
svmrs_fit <- train(data=dat_train_simple, DEFAULT~.,
                   method = "svmRadialSigma")
svmrs_predicted <- predict(svmrs_fit, dat_test)
svmrs_cm <- confusionMatrix(factor(svmrs_predicted),factor(dat_test$DEFAULT))

result_other <- rbind(result_other, with(svmrs_cm,
                                         c(Method = "svmRadialSigma",
                                           Accuracy = overall["Accuracy"],
                                           Balanced = byClass["Balanced Accuracy"],
                                           Sensitivity = byClass["Sensitivity"],
                                           Specificity = byClass["Specificity"],
                                           F1 = byClass["F1"])))
result_other %>% knitr::kable(caption="PREDICTION RESULT", digits=4)
```